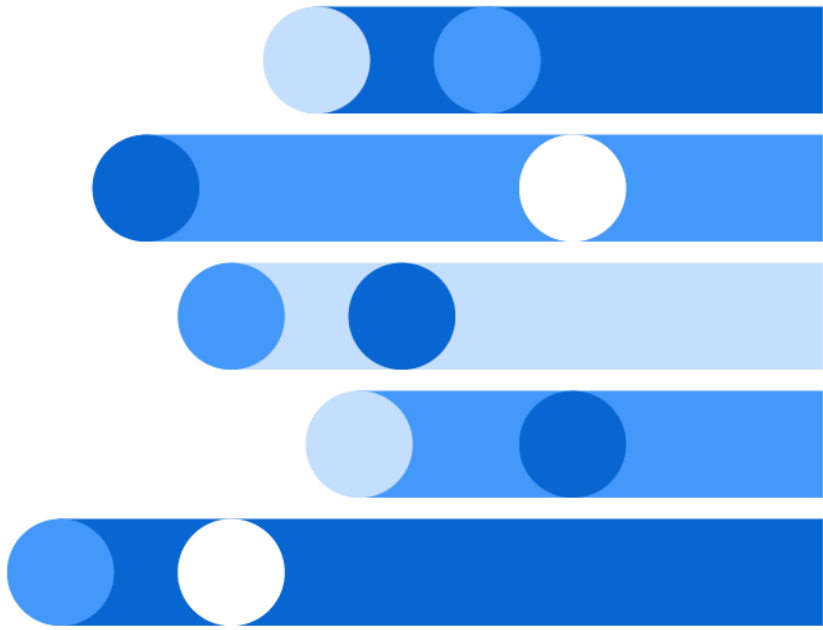
SAS® 9.4 Companion for Windows, Fifth Edition



SAS® Documentation
September 15, 2026

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS® 9.4 Companion for Windows, Fifth Edition*. Cary, NC: SAS Institute Inc.

SAS® 9.4 Companion for Windows, Fifth Edition

Copyright © 2016, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

September 2026

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

9.4-P10:hostwin

Contents

About This Book . . . *ix*
What's New in the SAS 9.4 Windows Companion . . . *xv*

PART 1 Running SAS under Windows 1

Chapter 1 / Getting Started under Windows . . . **3**
SAS: Exploiting the Power of Windows . . . 4
Starting SAS . . . 6
Files Used by SAS . . . 27
Submitting SAS Code . . . 41
Interrupting Your SAS Session . . . 44
Running Windows or MS-DOS Commands from within SAS . . . 45
Terminating a SAS Process . . . 49
Ending Your SAS Session . . . 50

Chapter 2 / Interacting with SAS under Windows . . . **53**
Overview of the SAS Interface . . . 54
Working within Your SAS Session . . . 57
Customizing Your SAS Session . . . 82
Accessing Online Help and Documentation . . . 102

Chapter 3 / Using the SAS Editors under Windows . . . **107**
Using the Enhanced Editor . . . 108
Using the Program Editor . . . 140

Chapter 4 / Using SAS Files under Windows . . . **151**
Introduction to SAS Files . . . 152
Multi Engine Architecture . . . 157
Using Data Libraries . . . 159
Accessing SAS Files from Multiple SAS Sessions . . . 171
Using SAS Files from Other Versions with SAS 9.4 for Windows . . . 172
Using SAS 9.4 Files with Previous Releases . . . 177
Using Remote Host SAS Files in SAS 9.4 . . . 178
Reading BMDP, OSIRIS, and SPSS Files . . . 178
Transferring SAS Files between Operating Environments . . . 183
Accessing Database Files with SAS/ACCESS Software . . . 183
Using the SAS ODBC Driver to Access SAS Data from Other Applications . . . 184

Chapter 5 / Using External Files under Windows . . . **185**
About External Files . . . 185

Referencing External Files 186
Accessing External Files with SAS Statements 198
Accessing External Files with SAS Commands 201
Advanced External I/O Techniques 204

Chapter 6 / Managing SAS Output under Windows **207**
Printing 208
Routing Procedure Output to a Web Browser 225
Routing Procedure Output and the SAS Log to a File 228
Using the SAS Logging Facility to Write Log Messages 231
Producing Graphics 232

Chapter 7 / Performance Considerations under Windows **245**
Hardware Considerations 245
Windows Features That Optimize Performance 247
SAS Features That Optimize Performance 252
Network Performance Considerations 254
Sasiotest Utility 254
Advanced Performance Tuning Methods 254

PART 2 Using SAS with Other Windows Applications 261

Chapter 8 / Using Windows System Tools with SAS under Windows **263**
Introduction to Using Windows System Tools with SAS 263
Event Viewer Application Log 264
Performance Tools 267
Starting SAS as a Windows Service 274

Chapter 9 / Using OLE in SAS/AF Software under Windows **285**
About OLE 286
SAS/AF Catalog Compatibility 286
Inserting an OLE Object in a FRAME Entry 287
Editing an OLE Object within a FRAME Entry 291
Invoking OLE Verbs 292
Using Linked OLE Objects 293
Converting OLE Objects 295
Automating OLE Objects and Applications 297
Using OLE Custom Controls (OCXs) in Your SAS/AF Application 305

Chapter 10 / Controlling SAS from Another Application Using OLE **315**
Introduction to Automating SAS 315
Creating an Instance of SAS 316
Getting Feedback from the SAS Session 316
Examples of Automating SAS with OLE 317
Methods and Properties for Use with a SAS OLE Automation Object 319
Dictionary 321

Chapter 11 / Using Unnamed and Named Pipes under Windows **325**
Overview of Pipes 325

Using Unnamed Pipes . . . 326
Using Named Pipes . . . 328

Chapter 12 / Accessing External DLLs from SAS under Windows . . . 339
Overview of Dynamic Link Libraries in SAS . . . 339
The SASCBTBL Attribute Table . . . 340
Special Considerations When Using External DLLs . . . 346
Examples . . . 358

Chapter 13 / Special Considerations for SAS/AF Programmers under Windows . . . 363
Controlling the Appearance and Behavior of SAS . . . 363
Controlling the Main SAS Window . . . 364
Accessing External DLLs from SAS . . . 368
Designing, Saving, and Loading Custom Toolbar Controls . . . 369
Invoking SAS/AF Applications Automatically . . . 370
Associating Your Own Logo and Icons with Your SAS/AF Application . . . 370

PART 3 Features of the SAS Language for Windows 373

Chapter 14 / Data Set Options under Windows . . . 375
SAS Data Set Options under Windows . . . 375
Dictionary . . . 375

Chapter 15 / SAS Commands under Windows . . . 379
SAS Commands under Windows . . . 381
Dictionary . . . 382

Chapter 16 / SAS Formats under Windows . . . 447
SAS Formats under Windows . . . 447
Writing Binary Data . . . 448
Accessing User-Written Formats from Releases Earlier Than SAS 9.4 . . . 449
Dictionary . . . 449

Chapter 17 / SAS Functions and CALL Routines under Windows . . . 461
SAS Functions and CALL Routines under Windows . . . 462
Dictionary . . . 462

Chapter 18 / SAS Informats under Windows . . . 497
SAS Informats under Windows . . . 497
Reading Binary Data . . . 498
Converting User-Written Informats from Earlier Releases to SAS 9.4 . . . 499
Dictionary . . . 500

Chapter 19 / SAS Procedures under Windows . . . 511
SAS Procedures under Windows . . . 511
Dictionary . . . 511

Chapter 20 / SAS Statements under Windows . . . 535
SAS Statements under Windows . . . 535
Dictionary . . . 536

Chapter 21 / SAS System Options under Windows **569**
SAS Options under Windows 572
Displaying SAS Option Settings 573
Changing SAS Option Settings 574
Processing System Options That Are Set in Several Places 577
SAS Options by Category 577
Dictionary 586

Chapter 22 / Length and Precision of Variables under Windows **725**
Length and Precision of Variables under Windows 725
Numeric Variables 725
Character Variables 727

Chapter 23 / SAS Macro Facility under Windows **729**
SAS Macro Facility under Windows 729
Automatic Macro Variables 730
Macro Statements 733
Macro Functions 734
Autocall Libraries 735

PART 4 Appendixes 737

Appendix 1 / Using Dynamic Data Exchange under Windows **739**
Dynamic Data Exchange (DDE) – A Legacy Method 740
Overview of DDE 742
DDE Syntax within SAS 743
Referencing the DDE External File 744
DDE Examples 746

Appendix 2 / Error Messages for SAS under Windows **755**
Overview of SAS Error Messages 755
Return Codes and Completion Status 756
Accessing Files 757
Using SAS Features 758
Using OLE 759
Using Networks 760
Resolving Internal Errors 761
Resolving Operating System and Windows Error Messages 762
Initialization and Termination Error Messages 762

Appendix 3 / Graphics Considerations under Windows **765**
Using TrueType Fonts with SAS/GRAPH Software 765

Appendix 4 / Default Key Settings for Interactive SAS Sessions under Windows **767**
Default Key Definitions under Windows 767
Keyboard Shortcuts within the SAS Main Window 769
Keyboard Shortcuts within the Enhanced Editor 772
Keyboard Shortcuts within Print Preview 776

Appendix 5 / Using EBCDIC Data on ASCII Systems **779**
About EBCDIC and ASCII Data 779
Moving Data from EBCDIC to ASCII Systems 782
Moving Data from ASCII to EBCDIC Systems 790

Appendix 6 / SCL Methods for Automating OLE Objects under Windows **795**
Summary of OLE Class Methods 795

Appendix 7 / Cleanwork Utility **807**
Cleanwork Utility 807
Scheduling the Cleanwork Utility with Microsoft Task Scheduler 808

Appendix 8 / SASIOTEST Utility **813**
Sasiotest Utility 813
Best Practices and Considerations 814
Acceptable SAS I/O Performance 815

About This Book

Syntax Conventions for the SAS Language

Overview of Syntax Conventions for the SAS Language

Overview of Syntax Conventions for the SAS Language

SAS uses standard conventions in the documentation of syntax for SAS language elements. These conventions enable you to easily identify the components of SAS syntax. The conventions can be divided into these parts:

- syntax components
- style conventions
- special characters
- references to SAS libraries and external files

Syntax Components

Syntax Components

The components of the syntax for most language elements include a keyword and arguments. For some language elements, only a keyword is necessary. For other language elements, the keyword is followed by an equal sign (=). The syntax for arguments has multiple forms in order to demonstrate the syntax of multiple arguments, with and without punctuation.

keyword
specifies the name of the SAS language element that you use when you write your program. Keyword is a literal that is usually the first word in the syntax. In a CALL routine, the first two words are keywords.

In these examples of SAS syntax, the keywords are bold:

CHAR (*string, position*)

CALL RANBIN (*seed, n, p, x*);

ALTER (*alter-password*)

BEST *w*.

REMOVE <*data-set-name*>

In this example, the first two words of the CALL routine are the keywords:

CALL RANBIN(*seed*, *n*, *p*, *x*)

The syntax of some SAS statements consists of a single keyword without arguments:

DO;
... *SAS code* ...
END;

Some system options require that one of two keyword values be specified:

DUPLEX | **NODUPLEX**

Some procedure statements have multiple keywords throughout the statement syntax:

CREATE <UNIQUE> **INDEX** *index-name* **ON** *table-name* (*column-1* <, *column-2*, ...>)

argument
specifies a numeric or character constant, variable, or expression. Arguments follow the keyword or an equal sign after the keyword. The arguments are used by SAS to process the language element. Arguments can be required or

optional. In the syntax, optional arguments are enclosed in angle brackets (< >).

In this example, *string* and *position* follow the keyword CHAR. These arguments are required arguments for the CHAR function:

CHAR (*string, position*)

Each argument has a value. In this example of SAS code, the argument *string* has a value of 'summer', and the argument *position* has a value of 4:

```
x=char('summer', 4);
```

In this example, *string* and *substring* are required arguments, whereas *modifiers* and *startpos* are optional.

FIND(*string, substring* <, *modifiers*> <, *startpos*>

argument(s)
specifies that one argument is required and that multiple arguments are allowed. Separate arguments with a space. Punctuation, such as a comma (,) is not required between arguments.

The MISSING statement is an example of this form of multiple arguments:

MISSING *character(s)*;

<LITERAL_ARGUMENT> *argument-1* <<LITERAL_ARGUMENT> *argument-2* ... >
specifies that one argument is required and that a literal argument can be associated with the argument. You can specify multiple literals and argument pairs. No punctuation is required between the literal and argument pairs. The ellipsis (...) indicates that additional literals and arguments are allowed.

The BY statement is an example of this argument:

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* ...>;

argument-1 <*options*> <*argument-2* <*options*> ...>
specifies that one argument is required and that one or more options can be associated with the argument. You can specify multiple arguments and associated options. No punctuation is required between the argument and the option. The ellipsis (...) indicates that additional arguments with an associated option are allowed.

The FORMAT procedure PICTURE statement is an example of this form of multiple arguments:

PICTURE name <(*format-options*)>
<*value-range-set-1* <(*picture-1-options*)>
<*value-range-set-2* <(*picture-2-options*)> ...>>;

argument-1=value-1 <*argument-2=value-2* ...>
specifies that the argument must be assigned a value and that you can specify multiple arguments. The ellipsis (...) indicates that additional arguments are allowed. No punctuation is required between arguments.

The LABEL statement is an example of this form of multiple arguments:

LABEL *variable-1=label-1* <*variable-2=label-2* ...>;

argument-1 <, *argument-2*, ...>
specifies that one argument is required and that you can specify multiple arguments that are separated by a comma or other punctuation. The ellipsis (...) indicates a continuation of the arguments, separated by a comma. Both forms are used in the SAS documentation.

Here are examples of this form of multiple arguments:

AUTHPROVIDERDOMAIN (*provider-1:domain-1* <, *provider-2:domain-2*, ...>

INTO :*macro-variable-specification-1* <, :*macro-variable-specification-2*, ...>

Note: In most cases, example code in SAS documentation is written in lowercase with a monospace font. You can use uppercase, lowercase, or mixed case in the code that you write.

Style Conventions

Style Conventions

The style conventions that are used in documenting SAS syntax include uppercase bold, uppercase, and italic:

UPPERCASE BOLD
identifies SAS keywords such as the names of functions or statements. In this example, the keyword ERROR is written in uppercase bold:

ERROR <*message*>;

UPPERCASE
identifies arguments that are literals.

In this example of the CMPMODEL= system option, the literals include BOTH, CATALOG, and XML:

CMPMODEL=BOTH | CATALOG | XML |

italic
identifies arguments or values that you supply. Items in italic represent user-supplied values that are either one of the following:

- nonliteral arguments. In this example of the LINK statement, the argument *label* is a user-supplied value and therefore appears in italic:

  LINK *label*;

- nonliteral values that are assigned to an argument.

  In this example of the FORMAT statement, the argument DEFAULT is assigned the variable *default-format*:

  FORMAT *variable(s)* <*format* > <DEFAULT = *default-format*>;

Special Characters

Special Characters

The syntax of SAS language elements can contain the following special characters:

=
: an equal sign identifies a value for a literal in some language elements such as system options.

  In this example of the MAPS system option, the equal sign sets the value of MAPS:

  MAPS=*location-of-maps*

< >
: angle brackets identify optional arguments. A required argument is not enclosed in angle brackets.

  In this example of the CAT function, at least one item is required:

  CAT (*item-1* <, *item-2*, ...>)

|
: a vertical bar indicates that you can choose one value from a group of values. Values that are separated by the vertical bar are mutually exclusive.

  In this example of the CMPMODEL= system option, you can choose only one of the arguments:

  CMPMODEL=BOTH | CATALOG | XML

...
: an ellipsis indicates that the argument can be repeated. If an argument and the ellipsis are enclosed in angle brackets, then the argument is optional. The repeated argument must contain punctuation if it appears before or after the argument.

  In this example of the CAT function, multiple *item* arguments are allowed, and they must be separated by a comma:

  CAT (*item-1* <, *item-2*, ...>)

'*value*' or "*value*"
: indicates that an argument that is enclosed in single or double quotation marks must have a value that is also enclosed in single or double quotation marks.

  In this example of the FOOTNOTE statement, the argument *text* is enclosed in quotation marks:

  FOOTNOTE <*n*> <*ods-format-options* '*text*' | "*text*">;

;

a semicolon indicates the end of a statement or CALL routine.

In this example, each statement ends with a semicolon:

```
data namegame;
   length color name $8;
   color = 'black';
   name = 'jack';
   game = trim(color) || name;
run;
```

References to SAS Libraries and External Files

References to SAS Libraries and External Files

Many SAS statements and other language elements refer to SAS libraries and external files. You can choose whether to make the reference through a logical name (a libref or fileref) or use the physical filename enclosed in quotation marks.

If you use a logical name, you typically have a choice of using a SAS statement (LIBNAME or FILENAME) or the operating environment's control language to make the reference. Several methods of referring to SAS libraries and external files are available, and some of these methods depend on your operating environment.

In the examples that use external files, SAS documentation uses the italicized phrase *file-specification*. In the examples that use SAS libraries, SAS documentation uses the italicized phrase *SAS-library* enclosed in quotation marks:

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```

What's New in the SAS 9.4 Windows Companion

Overview

SAS under Windows enables you to complete your data and computation tasks in SAS while integrating with the Windows applications that are already in place on your desktop, within your desktop, and within your enterprise. SAS supports information sharing through Windows tools and techniques.

General Enhancements

The following enhancements are implemented for SAS 9.4:

- The content in "Choosing a Location for the Sorted File" on page 255 has been modified to better explain the sort actions.
- References to the GIMPORT procedure have been removed. GIMPORT does not work with the new TrueType fonts.
- An appendix, Troubleshooting Java Runtime Environment Errors, has been removed. The instructions are in the SAS note that is located at http://support.sas.com/kb/48/548.html.
- Extended attributes are described in "SAS Data Sets and Data Views (Member Types: Data and View)" on page 154.
- The default value of the LRECL= option is 32767.
- A chapter, "Using Lotus Notes to Distribute SAS Data under Windows," has been removed. Lotus Notes is not supported on WX6.
- Windows XP, Windows Server 2003, and Windows Vista are no longer supported. Supported Windows Editions are listed in "SAS Runs in Enterprise Environments" on page 4.

- In SAS 9.4M1, details have been added in support of Windows 32-bit architecture.
- In SAS 9.4M1, the accessibility topic was removed. The revised accessibility topic is available on the Base product page.
- In SAS 9.4M1, an appendix "Sasiotest Utility" on page 813 has been added.
- In SAS 9.4M2, Appendix 5, "Using EBCDIC Data on ASCII Systems," on page 779 has been added.

In SAS 9.4M4, the following enhancements have been made:

- for SAS 9.4, the .spds9 file extension was added to "File Extensions for SAS Files" on page 152.
- the "Safe save (Enhanced Editor only)" on page 87 option is new in the Edit Preferences window.

In SAS 9.4M5, Windows 2016 is supported.

In the November 2019 release of SAS 9.4M6, Microsoft Windows Server 2019 is supported.

Documentation Enhancements

The documentation for the TOOLSMENU and VIEWMENU system options is now located in the *SAS Companion for Windows*. The documentation was removed from the *SAS System Options: Reference*.

Cleanwork Utility

In SAS 9.4M2, the Cleanwork utility was updated. It is now a console-based application and is installed as part of Base SAS in the SASROOT directory. For more information, see "Cleanwork Utility" on page 807.

In SAS 9.4M4, this note was added: The Cleanwork utility is available beginning with SAS 9.4M2.

Dynamic Data Exchange

In SAS 9.4M5, a chapter, "Dynamic Data Exchange (DDE) – A Legacy Method" was moved to the Appendix. DDE is a legacy method and should not be used with modern processes.

Functions

In SAS 9.4M1, the following interaction was added to the PEEKLONG function: "When a SAS server is in a locked-down state, the PEEKLONG function does not execute." For more information, see "PEEKLONG Function: Windows" on page 489.

Statements

In SAS 9.4M2, the PERMISSION option was added to the following statements: FILE statement on page 538 and FILENAME statement on page 542.

In SAS 9.4M2, the following FILENAME statement access methods are not available when SAS is in a locked-down state. For more information, see the FILENAME statement on page 542.

- EMAIL
- FTP
- Hadoop
- SOCKET
- URL

However, your server administrator can re-enable the access method so that it is accessible in the locked-down state.

System Options

The Summary of System Options table has been replaced with the SAS System Options by Category table on page 577.

The following system options are new:

ALIGNSASIOFILES
: Aligns the pages of data in a SAS data set.

FILELOCKWAIT
: Sets the number of seconds that SAS waits for a locked file to become available.

HOSTINFOLONG
: Specifies to write additional operating environment information in the SAS log when SAS starts.

In SAS 9.4M2, the following tip was added to the RTRACELOC system option: "You can expand the RTRACELOC filename when %p (pid), %d (date), %t (time) are specified." For more information, see "RTRACELOC System Option: Windows" on page 674.

In SAS 9.4M2, the default value was changed to 1G in the SORTSIZE option. For more information, see "SORTSIZE System Option: Windows" on page 691.

In SAS 9.4M3, the default value for the MVARSIZE system option was changed to 65534 bytes. For more information, see "MVARSIZE System Option: Windows" on page 651.

PART 1

Running SAS under Windows

Chapter 1
Getting Started under Windows 3

Chapter 2
Interacting with SAS under Windows 53

Chapter 3
Using the SAS Editors under Windows 107

Chapter 4
Using SAS Files under Windows 151

Chapter 5
Using External Files under Windows 185

Chapter 6
Managing SAS Output under Windows 207

Chapter 7
Performance Considerations under Windows 245

1

Getting Started under Windows

SAS: Exploiting the Power of Windows **4**
- SAS Runs in Enterprise Environments 4
- An Integral Part of Your Windows Editions 4
- Compatible and Maintainable 5
- Launching Java 5

Starting SAS **6**
- Use SAS Interactively or in Batch Mode 6
- Starting from the Start Menu in the SAS Windowing Environment 6
- Starting from Custom Shortcuts or Program Items 7
- Starting from the Run Dialog Box or a Command Prompt 8
- Starting from a SAS File 9
- Running SAS in Batch Mode 9
- Starting the Program Editor When SAS Starts 23
- Determining the Current Folder When SAS Starts 24
- Sample SAS Session 24
- What If SAS Does Not Start? 27

Files Used by SAS **27**
- Introduction to Files Used by SAS 27
- SAS Configuration Files 28
- SAS Autoexec File 34
- Profile Catalog 36
- Work Data Library 38
- SAS Registry Files 39
- SAS Default Folder Structure 40

Submitting SAS Code **41**
- Introduction to Submitting SAS Code 41
- Submitting Code from the Enhanced Editor or Program Editor 41
- Submitting Code from the SAS NOTEPAD Text Editor 42
- Submitting Code from the Clipboard 42
- Submitting Code By Dragging and Dropping 43
- Submitting Code Stored in Registered SAS File Types 44

Interrupting Your SAS Session **44**

Running Windows or MS-DOS Commands from within SAS **45**
- Overview of Running Windows or MS-DOS Commands from within SAS 45
- Running Windows Commands Using the X Statement or the X Command 46
- Using a DATA Step to Issue Conditional Operating System Commands Conditionally 47

XWAIT System Option 47
XSYNC System Option 48
Comparison of the XWAIT and XSYNC System Options 49
Terminating a SAS Process **49**
Ending Your SAS Session **50**

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.

You can access applications that are supported by Dynamic Data Exchange (DDE) by accessing the following URL. Word and Excel applications are supported. The PowerPoint application is not supported.

Note: DDE a legacy method and should not be used with modern processes.

An Integral Part of Your Windows Editions

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. Those tools and techniques include the following:

- OLE
- Dynamic Data Exchange (DDE)
- Open Database Connectivity (ODBC)
- email system
- 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, you must convert catalogs from earlier formats (such as Release 9.1 under Windows) to SAS 9.4 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.

Launching Java

SAS launches Java using the Java Runtime Environment (JRE) so that it can run Java code. Parts of SAS are written in Java. The JRE contains the libraries that are needed to run Java.

In SAS 9.3, the ODS graphics procedures moved from SAS/GRAPH to Base SAS. These procedures that rely on Java are delivered as part of Base SAS, rather than as a separate SAS product. As a result, a key difference in the SAS 9.4 windowing environment is that SAS now initializes the Java environment upon SAS invocation.

HTML is now the default output for SAS 9.4 Windowing environment, replacing the listing destination. The default style has changed from Styles.Default to Styles.HTMLBlue for the default HTML ODS destination that is generated with the SAS windowing environment and in batch mode.

Java is implemented as a part of Base SAS in SAS 9.4 because the Graph Template Language and the SAS/GRAPH procedures such as SGPLOT, SGPANNEL, SGSCATTER, and SGRENDER have moved to Base SAS from the SAS/GRAPH procedures. By default, the ODS GRAPHICS statement is set to ON for SAS procedures that support ODS Graphics when the procedures are executed in the SAS windowing environment in the Windows and UNIX operating systems.

Although ODS Graphics are enabled for these procedures, you might want to use the ODS GRAPHICS statement to control the graphical environment.

When you use the Windowing environment of SAS (interactive SAS) in SAS 9.4, you might encounter error messages similar to the following in the SAS Log window when you run programs (for example: PROC JAVAINFO; run;):

```
ERROR: Unable to attach current thread"
   ERROR: Unable to load the java virtual machine"
   ERROR: Proc javainfo did not run correctly
   ERROR: The Java proxy is not responding.
   ERROR: The Java proxy's JNI call to start the VM failed.
   ERROR: Java failed to start during the SAS startup.
```

These errors are typically related to issues with the Java Runtime Environment (JRE). To resolve these issues, follow these steps: http://support.sas.com/kb/48/548.html

Starting SAS

Use SAS Interactively or in Batch Mode

When running SAS under Windows, or SAS Windowing environment, you can start an interactive session to submit programs and view the resulting output, or you can execute batch SAS jobs, and view the output 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 **Do not show this dialog box again**. You can start the tutorial at any time by selecting **Help** ⇨ **Getting Started with SAS Software.**

You can also start SAS with SAS Studio. See SAS Studio Overview for information about using SAS Studio.

Starting from the Start Menu in the SAS Windowing Environment

To start SAS from the Windows Start Menu:

1. Click **Start**.
2. Select **Programs**.
3. Select **SAS**.
4. Select **SAS 9.4 (Language)** .

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 Start up 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 append system options:

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 pathnames. The following example uses double quotation marks.

   For SAS 9.4:

   ```
   "c:\program files\SASHome\SASFoundation\9.4\sas.exe" -config "c:\mydir\sasv9.cfg"
   ```

   For SAS Viya:

   ```
   "c:\program files\SASHome\SASFoundation\9.4\sas.exe" -config "c:\mydir\sasv9.cfg"
   ```

   Note: If a system option's value has a space in it, you must enclose the value in quotation marks on the command line or in a config. file. The following example shows the correct syntax:

   ```
   -bottommargin '2 in';
   ```

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 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 31 .

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 36 . For more information about the Work data library, see "Work Data Library " on page 38 .

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, enter 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\SASHome\SASFoundation\9.4`**, you enter `"c:\program files\SASHome\SASFoundation\9.4\sas.exe"`, and the options that you want to specify.

Note: For Windows, you must specify a fully qualified path to sas.exe in the **Start Menu Search** entry field.

Using the SAS Command from the Command Prompt

To start an interactive SAS session, enter the SAS command from the command prompt. For example, the following code starts an interactive session, specifies the line size and page size, and indicates the location of the SAS configuration file.

```
"c:\program files\SASHome\SASFoundation\9.4\sas.exe" -ls 80 -ps 60
   -config "c:\program files\SASHome\SASFoundation\9.4\sasv9.cfg"
```

The code works for both SAS 9.4 and SAS Viya.

Likewise, to start a batch SAS job, enter the SAS command at the command prompt. The following code starts a batch SAS job and indicates the location of the SAS configuration file.

```
"c:\program files\SASHome\SASFoundation\9.4\sas.exe"
   -sysin c:\mysas\programs\prog1.sas
   -config "c:\program files\SASHome\SASFoundation\9.4\sasv9.cfg"
```

The code works for both SAS 9.4 and SAS Viya.

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.

Running SAS in Batch Mode

Overview of Running SAS in Batch Mode

You can run SAS jobs in batch mode in the Windows operating environment. Place your SAS statements in a file and submit them for execution along with the control statements and system commands that are required at your site.

You can also run SAS jobs in batch mode in SAS Studio using the background job submission preferences. See Setting Preferences for Background Job Submissions for information about submitting batch jobs in SAS Studio.

For SAS Viya, you must access the CAS server. For more information about CAS, see An Introduction to SAS Viya Programming. This link provides information about how to access a CAS server and SAS Viya.

Submitting a Batch SAS Job

Note: The maximum line length is 32767 bytes.

You can submit a batch SAS job by using the following methods:

- For SAS 9.4: specify the SYSIN system option in the SAS command (issued from the command prompt or in the **Run** dialog box) and specify the SAS program to submit. Here is an example: `"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" -sysin c:\SASPrograms\prog1.sas -config "c:\Program Files\SASHome\SASFoundation\9.4\sasv9.cfg".`

  For SAS Viya: `"c:\Program Files\SAS\Viya\9.4\Sas.exe" -sysin c:\SASPrograms\prog1.sas -config "c:\Program Files\SASHome\SASFoundation\9.4\sasv9.cfg"`
- Right-click to select a file that has either a .sas, .ss2, .ss7, or .sas7bpgm file extension. From the pop-up menu, select **Batch Submit with SAS 9.4**.
- Select and drag your SAS program file icon (for the file that contains the SAS code) in Windows Explorer, and drop the file onto the SAS.EXE file icon or shortcut.

  Note: If you want to establish a permanent libref, specify the STARTLIB system option when you begin the batch job.

  Note: The -NOSTATUSWIN option enables you to run SAS in batch mode so that no windows are displayed. You can add options such as -NOTERMINAL, -NOSPLASH, -NOSTATUSWIN, and -NOICON to prevent the windows from being displayed.

The Status Window

When you use 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 available 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; 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.

Canceling a Batch Job

You can cancel a batch job by using the keyboard or the mouse:

- press CTRL+BREAK.
- click **Cancel** in the status window.

Running Windowing Procedures in a Batch Job

You can run windowing procedures in a batch job along with SAS/GRAPH, SAS/INSIGHT, and SAS/ACCESS software. When SAS reaches a point in your program where interaction is required, the main SAS window appears.

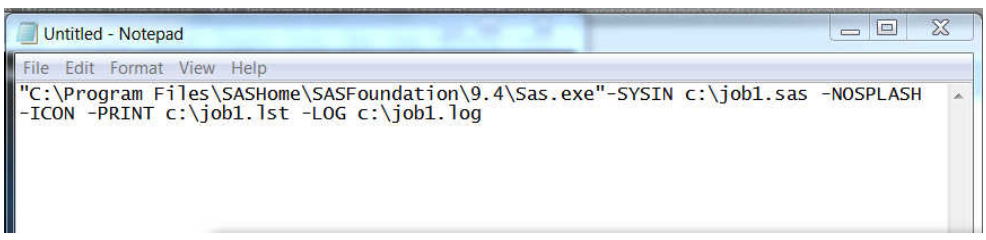
The following examples show how to execute SAS batch jobs under Windows 7.

Example 1: Creating a Batch File in Notepad or Another Text Editor

This example uses a BAT file. A BAT file is executed by the operating system.

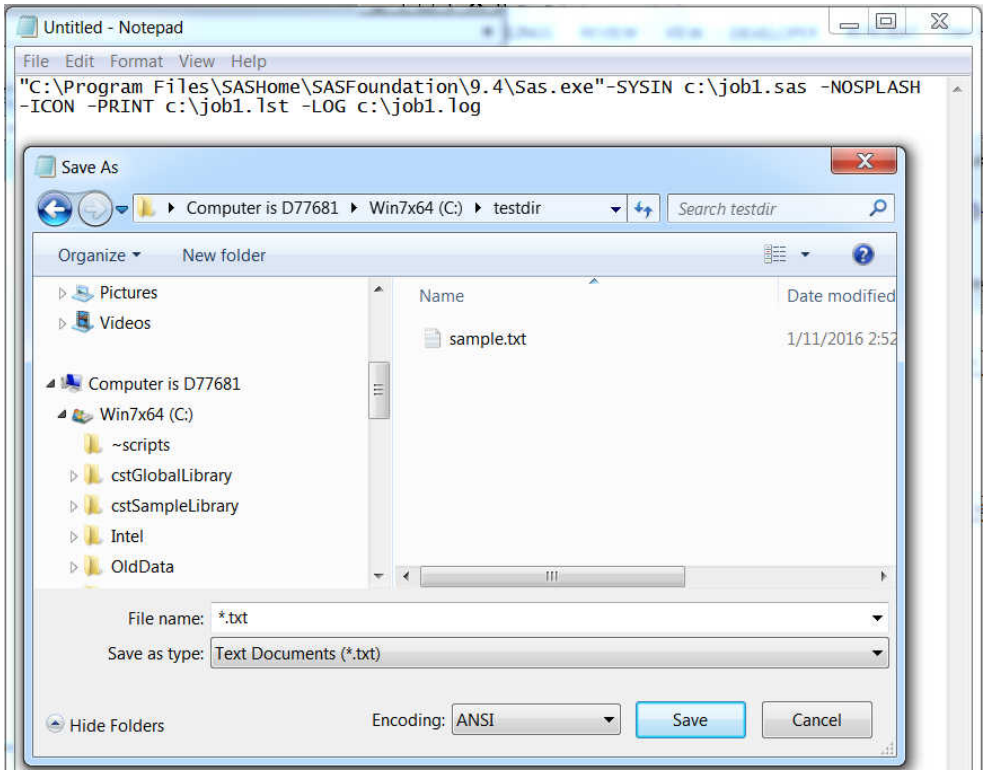
1. Create a file in Notepad or a similar text editor.
2. Enter a command similar to the following: `"C:\Program Files\SASHome\SASFoundation\9.4\Sas.exe"-SYSIN c:\job1.sas -NOSPLASH -ICON -PRINT c:\job1.lst -LOG c:\job1.log`

Figure 1.1 *Entering a Command in a BAT File*



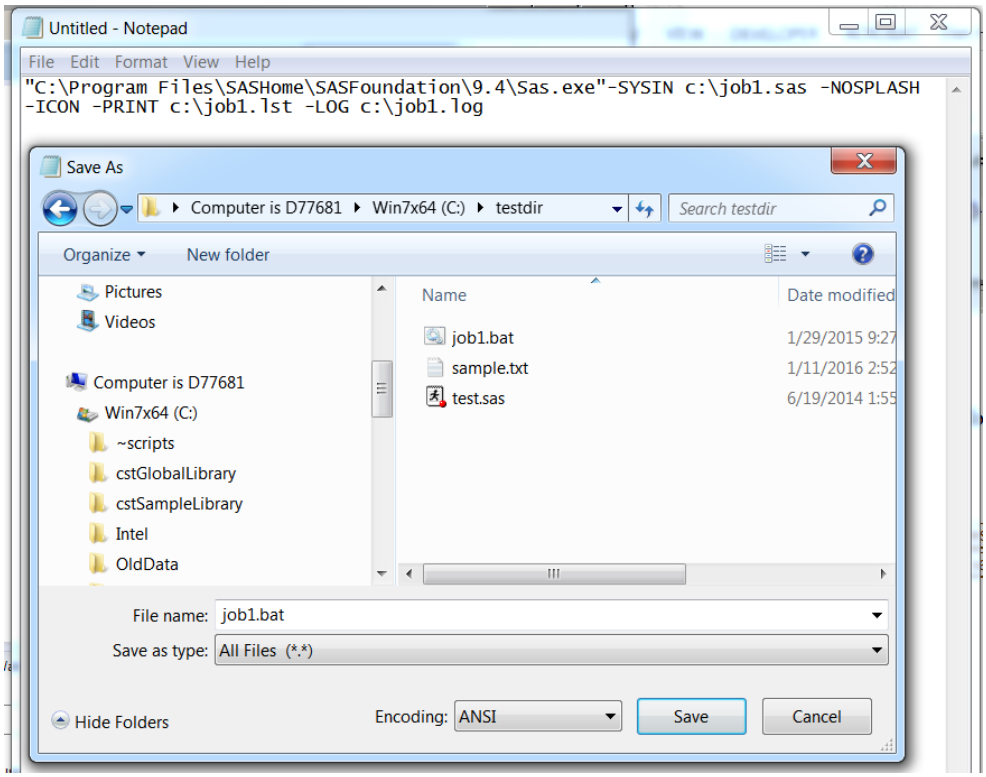
3. Select **File** ⇨ **Save As**.

Figure 1.2 *Saving a Batch File*



4 The default file extension is .txt. You must change the extension to .bat in order for other programs to access the file.

Figure 1.3 *Changing the File Extension to .bat*



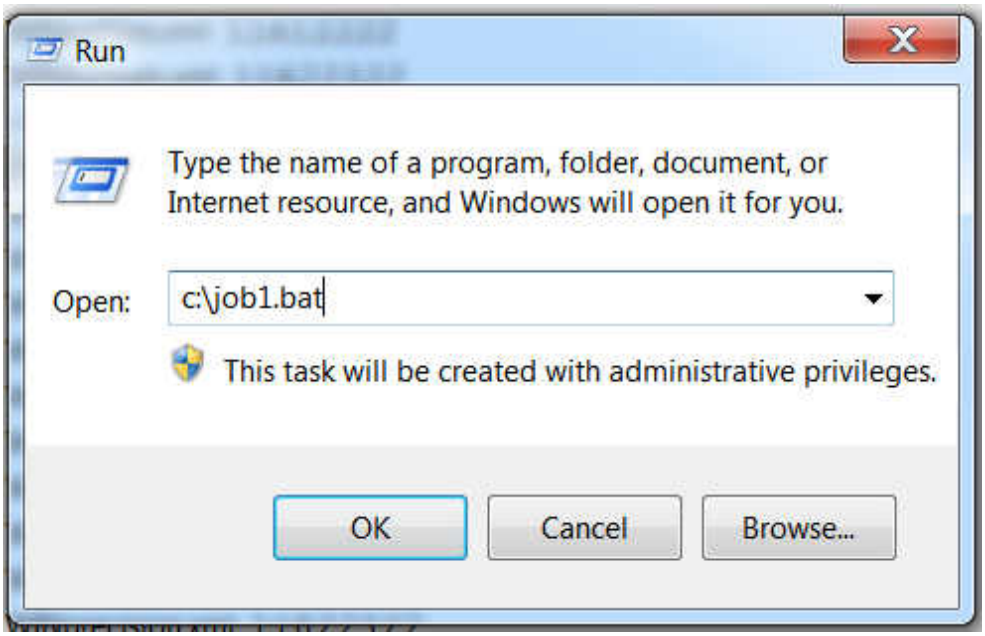
5 To execute the BAT file, select **Start** ⇨ **Run** or double-click the BAT file.

The resulting LOG file and the LST files reside in the same folder where the BAT file resides. To change the destination of the LOG and LST files, use the -LOG and -PRINT options.

Use the -PRINT option to change the destination folder for the output of the program. Use the -LOG option to change the destination folder for the log of the program. Here is an example: `"C:\Program Files\SASHome\SASFoundation\9.4\Sas.exe"-SYSIN c:\job1.bat -NOSPLASH -ICON -PRINT c:\job1.lst -LOG c:\job1.log`

Note: When you run a BAT program, a DOS window appears and remains available until the job is finished.

Figure 1.4 *Submitting a BAT Program*



Example 2: Submitting Batch Files from Explorer

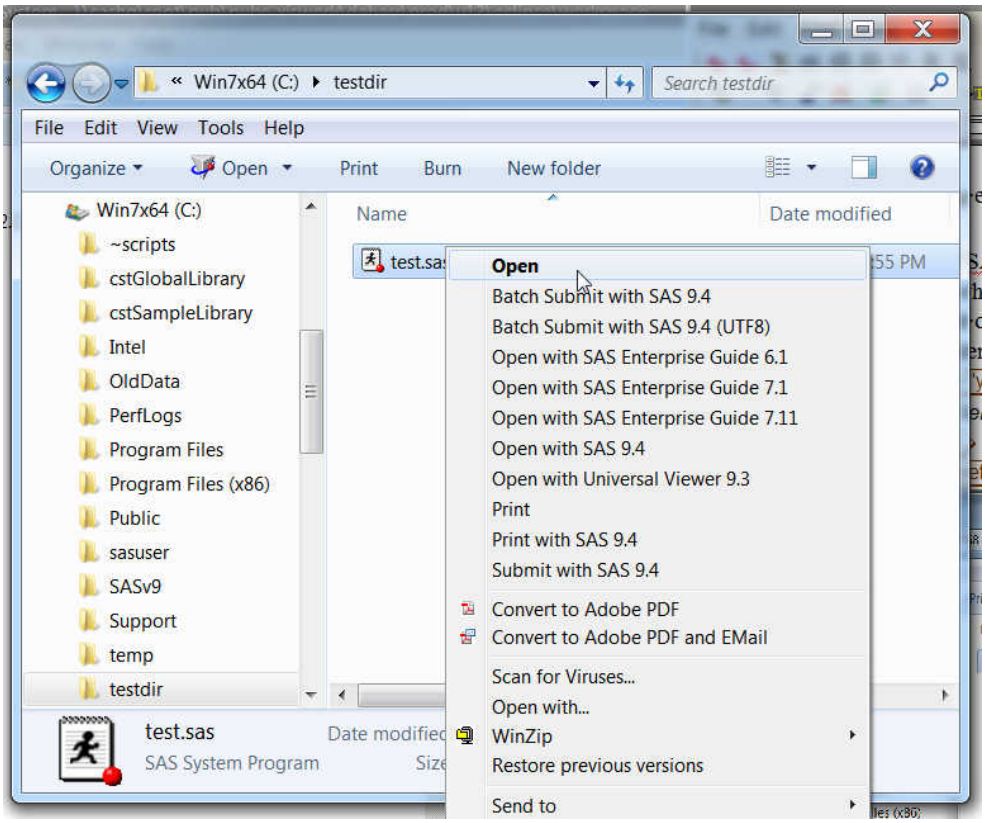
1 Open Explorer.

2 Right-click the program.

3 Select **Batch Submit with SAS 9.4**.

Note: By default, the LOG and LST files are located in the same folder as the program.

Figure 1.5 *Selecting the Batch Submit Option*



The LOG file is the log for the program.

The LST file is the output file for the program. This file is created only if there is output for the program.

Note: SAS must be properly installed before you can use batch processing. For information about SAS installation, see the chapter about SAS Deployment Manager tasks in SAS Deployment Wizard and SAS Deployment Manager 9.4: User's Guide.

You can also run SAS files in batch when you double-click the files within Explorer. The default action for SAS files must be set to `Batch Submit`.

To change the default action for SAS files:

1. Open Explorer.
2. Select a SAS file.
3. Right-click **Open With** and select **Default program**.
4. Select **SAS System for Windows**.
5. Double-click a SAS file to run the SAS program in batch mode. The LOG and LST files are created in the same folder as the SAS file.

Example 3: Running Multiple Programs within the Batch File

The following example runs five programs consecutively:

```
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test1.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test2.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test3.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test4.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test5.sas
```

Example 4: Using the SAS Executable Command with the -SYSIN Option

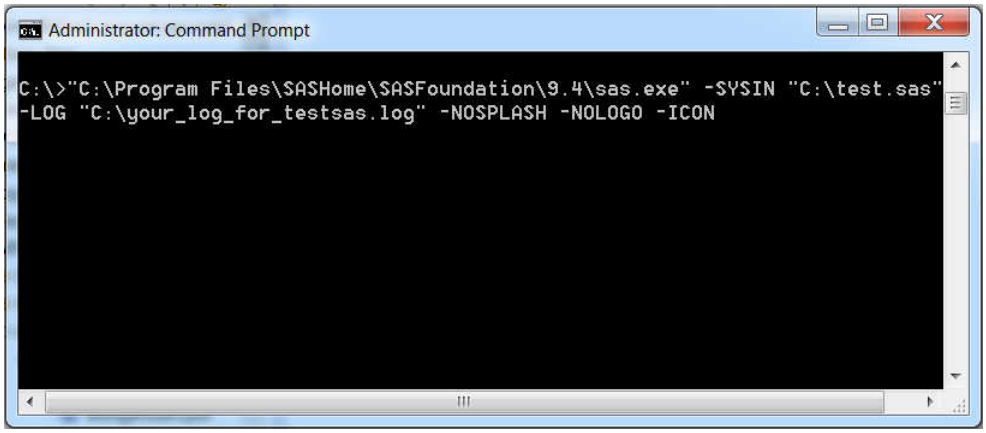
1 Select **Start** ⇨ **Run**.

2 Enter a command similar to the following: `"c:\Program Files\SASHome \SASFoundation\9.4\Sas.exe" -sysin "c:\test.sas" -LOG "c:\your_log_for_testsas.log" -NOSPLASH -NOLOGO -ICON`. Specify the location of the program. This example runs the TEST.SAS program that is located in the root of the C drive. Use the –NOSPLASH option to eliminate the splash screen. Add the –ICON option to minimize the DOS window when the program is started. The resulting LOG and LST files reside in the SAS root directory.

   The –SYSIN option specifies the SAS program file that runs in batch. The path must be a valid Windows path.

   Note: Use the –LOG option with the –SYSIN option when scheduling a BAT file to prevent the scheduler from writing the LOG file to an unexpected location.

Figure 1.6 *Using the –LOG and –SYSIN Options*



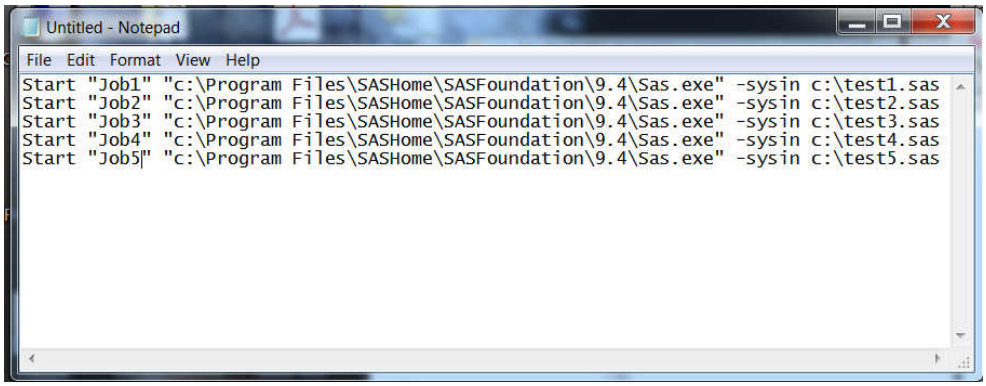
Example 5: Using the START Command to Run Concurrent Batch Jobs

The START command requires a title. The title must be in double quotation marks. This example uses Job# as the title.

Enter commands similar to the following:

```
Start "Job1" "c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe"
–sysin c:\test1.sas
Start "Job2" "c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe"
–sysin c:\test2.sas
Start "Job3" "c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe"
–sysin c:\test3.sas
Start "Job4" "c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe"
–sysin c:\test4.sas
Start "Job5" "c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe"
–sysin c:\test5.sas
```

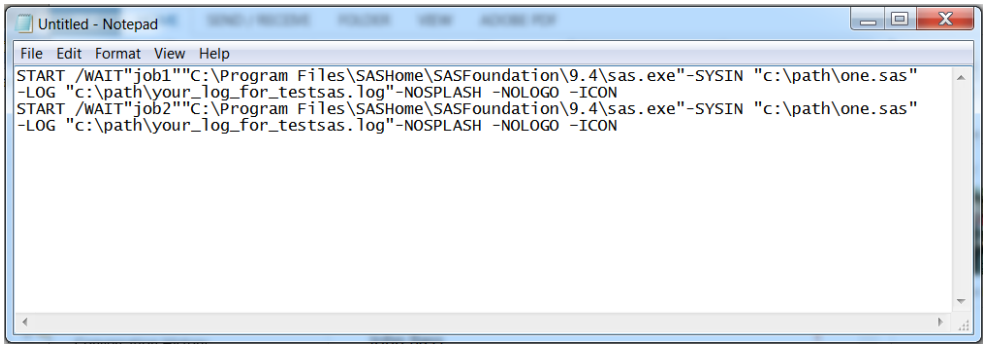
Figure 1.7 *Running Concurrent Batch Jobs Using the START Command*



Example 6: Running Jobs Consecutively Using the START and WAIT Commands

When you use the START and WAIT commands together, the order is as follows: the first job runs. After the first job is complete, the second job runs. In this example, Job1 runs first, and then Job2 runs after Job1 is complete.

Figure 1.8 *Using the START and WAIT Commands*



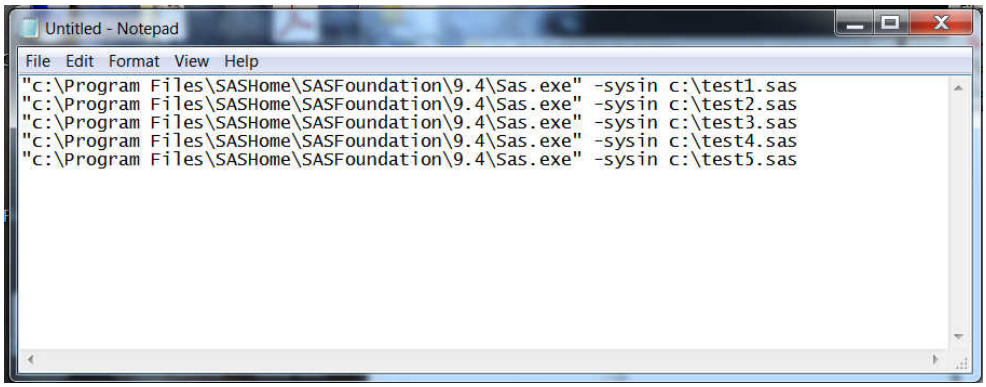
Example 7: Creating a Batch File That Runs Programs Consecutively

The log (LOG) and output (LST) files are located in the directory where the BAT file is located unless you use the –PRINT and –LOG options on the command line.

Enter the following commands in the BAT file:

```
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test1.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test2.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test3.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test4.sas
"c:\Program Files\SASHome\SASFoundation\9.4\Sas.exe" –sysin
c:\test5.sas
```

Figure 1.9 Running Concurrent Batch Jobs



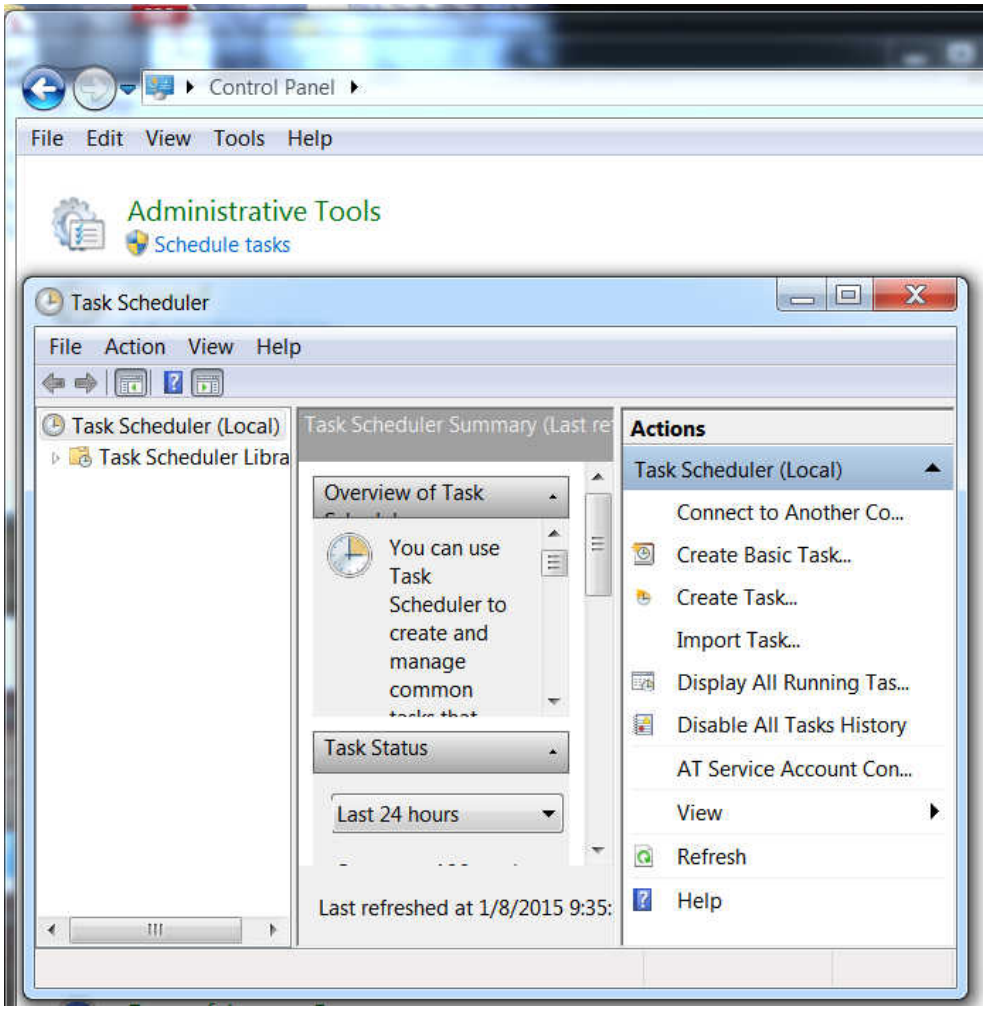
Example 8: Using the Task Scheduler to Run SAS Jobs

When jobs are scheduled, the programs can be executed without operator assistance. The **Schedule Tasks** program is located in the **Administrative Tools** section in the **Control Panel** under Windows.

In this example, job1.bat is the file that you schedule with Task Scheduler.

1 Click the **Schedule Tasks** icon to start the **Task Scheduler Wizard**.

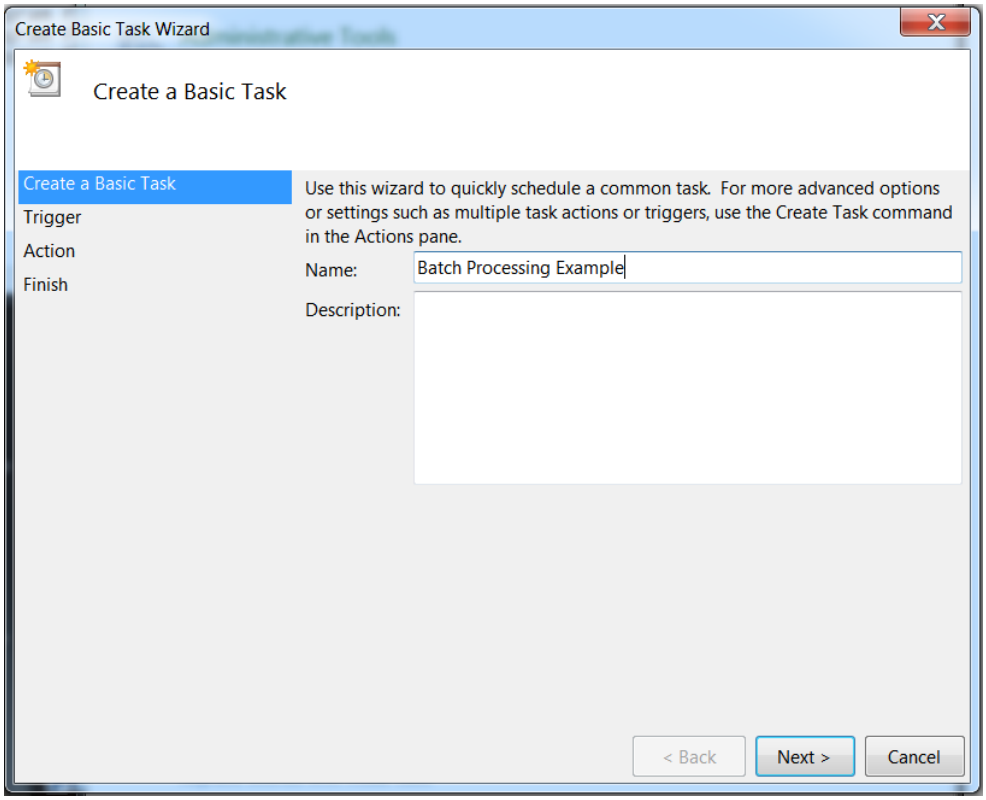
Figure 1.10 *Batch Processing with the Task Scheduler*



2 Select **Create Basic Task** or **Create Task**.

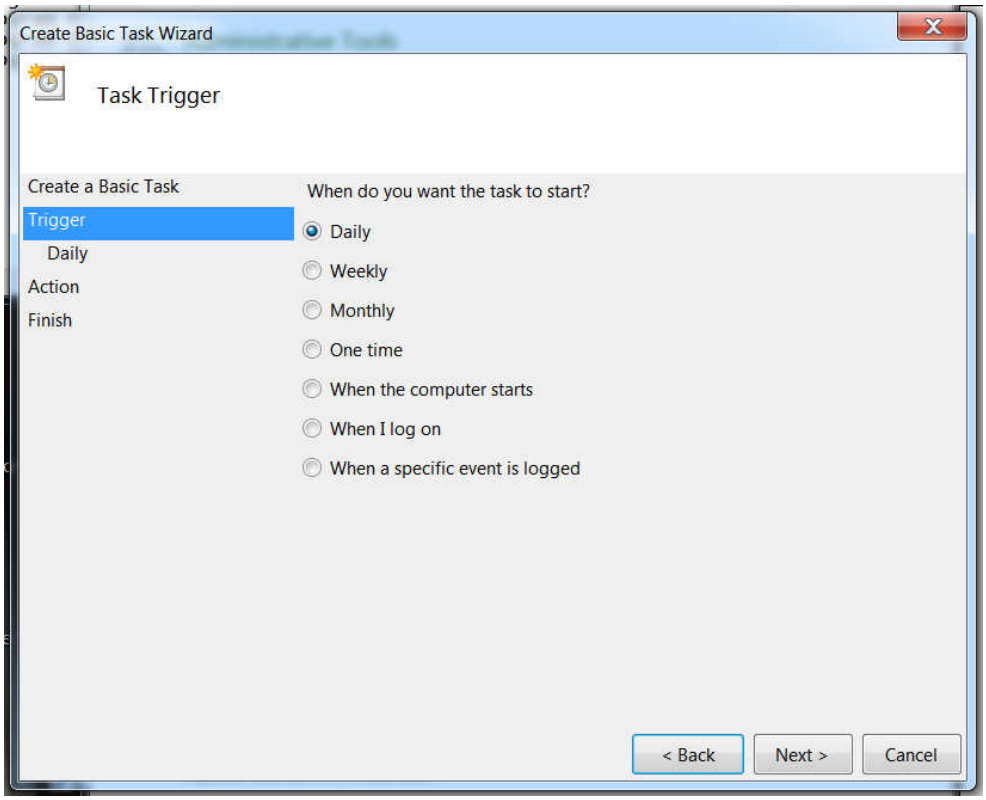
3 Enter the name of the task in the **Name** field. You can enter a description of the task in the **Description** field. Select **Next**.

Figure 1.11 *Creating a Basic Task for Batch Processing*



4 Select the time at which you want the task to start. Select **Next**.

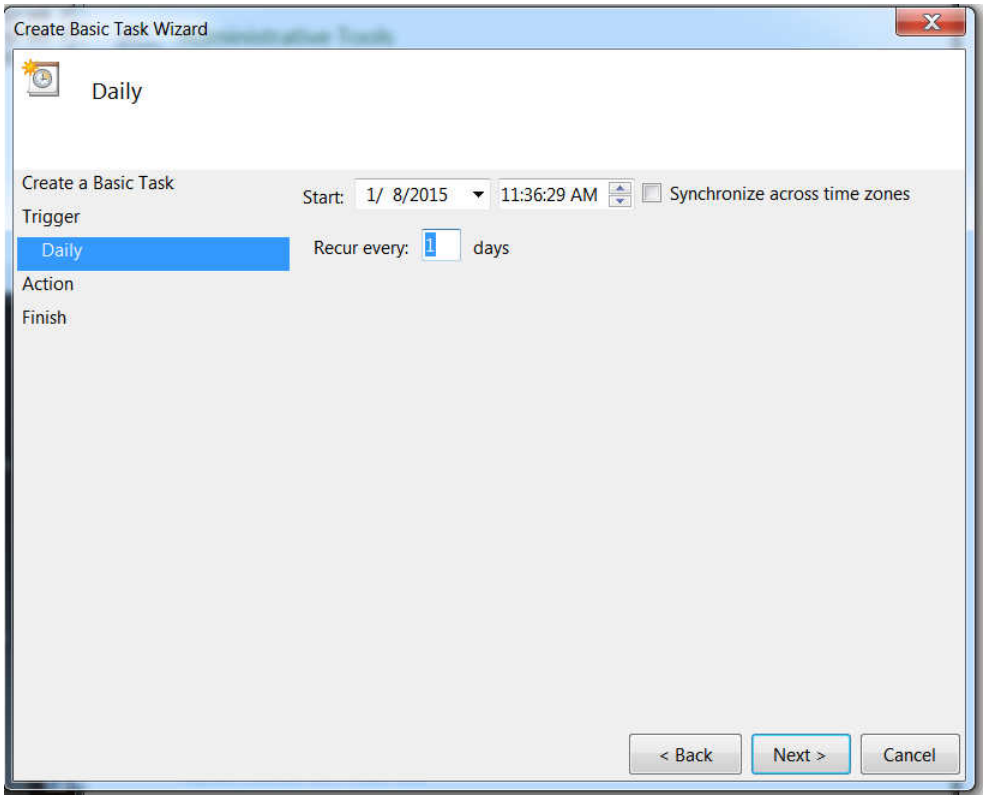
Figure 1.12 *Task Trigger*



5 Enter the user name and password that apply to this program. Select **Next**. A confirmation message is displayed.

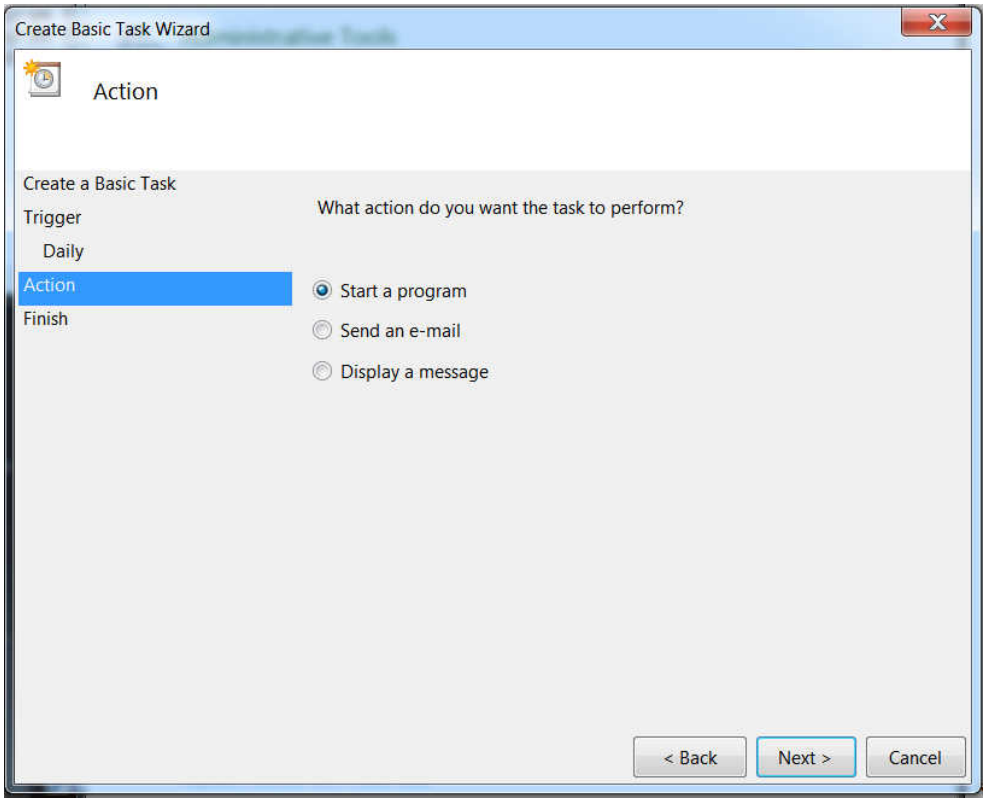
6 Specify the date and the time that you want the task to begin. Select **Next**.

Figure 1.13 *Processing a Daily Task*



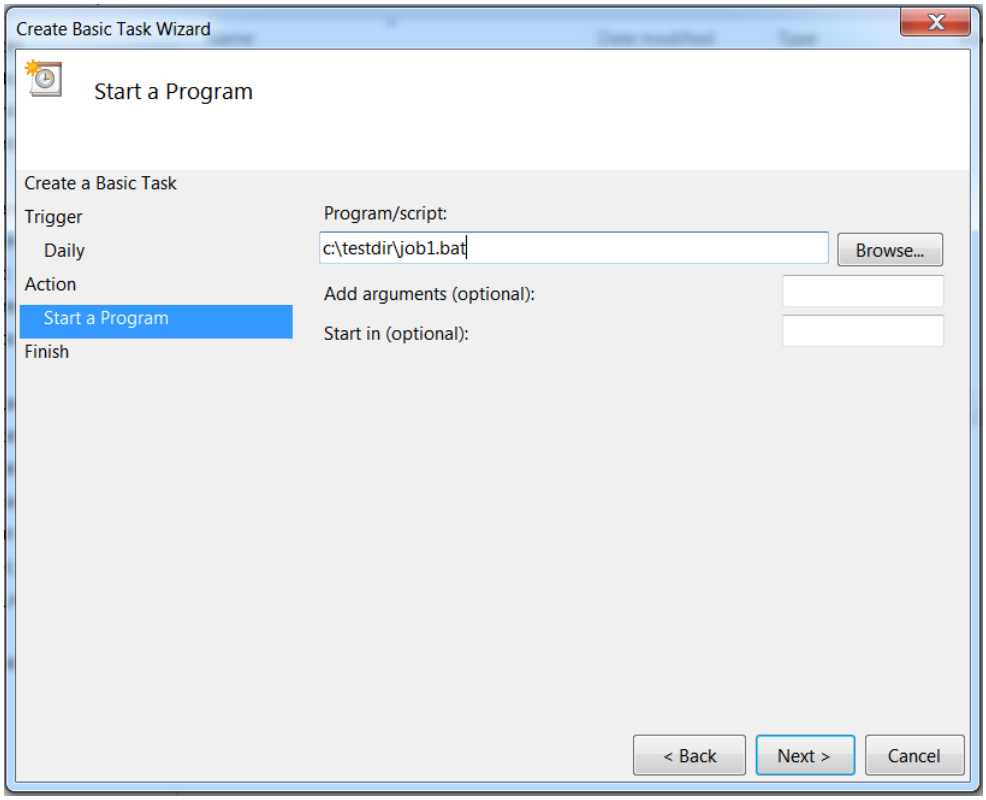
7 Select **Start a program** and select **Next**.

Figure 1.14 *Selecting Start a Program*



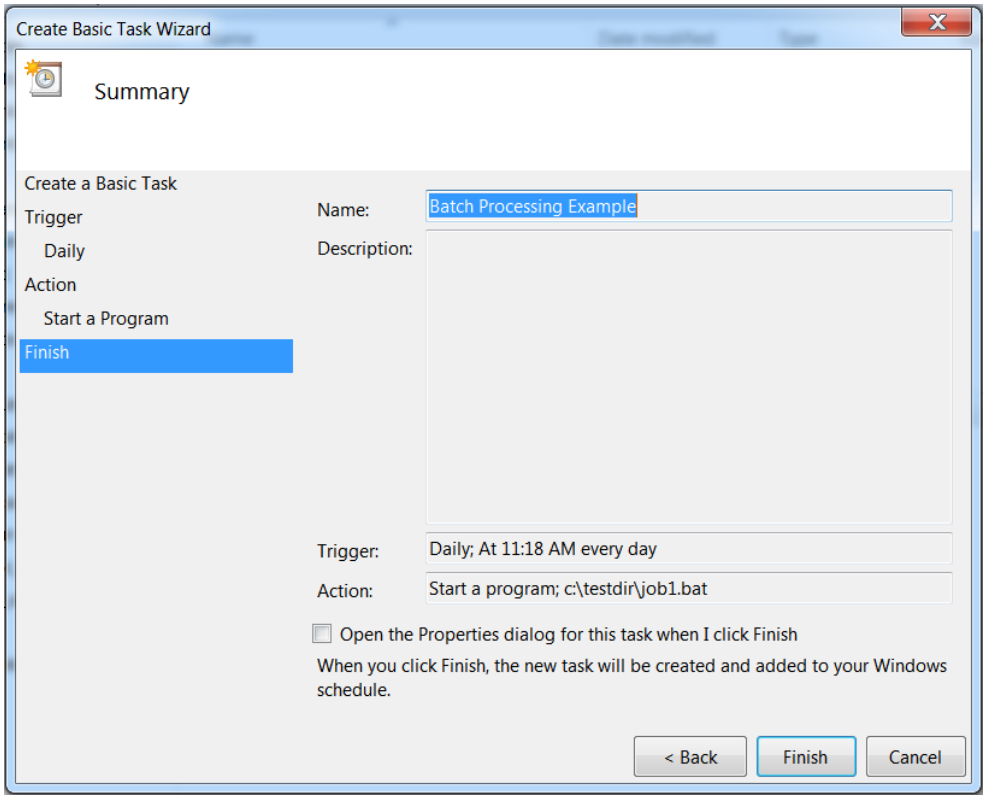
8 Click **Browse** and select the program. Select **Next**.

Figure 1.15 *Selecting Browse*



9 Review your selections.

Figure 1.16 *Create Basic Task Summary*



10 Select **Finish**.

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 140 , "Edit Preferences" on page 86 , and "ENHANCEDEDITOR System Option: Windows" on page 612 .

Determining the Current Folder When SAS Starts

By default, SAS determines the current folder. SAS uses the current folder as the location to read and write SAS files when you do not specify a different pathname.

SAS also searches the current folder, based on the following statements, for the AUTOEXEC.SAS file or INITSTMT files. In this case, the path that the SASINITIALFOLDER system option specifies is disregarded.

However, you can specify a pathname to use for the current folder by using the SASINITIALFOLDER system option when you start SAS. Alternatively, you can use the following rules to determine the current folder:

1. If you use a program item or shortcut to start SAS and if 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 if the command contains a path to the SAS.EXE file, 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 if you do not specify a path as part of the SAS command, then the current folder is specified by 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
```

For more information, see "Changing the SAS Current Folder" on page 60 and "SASINITIALFOLDER System Option: Windows" on page 679 .

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 38 .

Sample SAS Session

This section illustrates

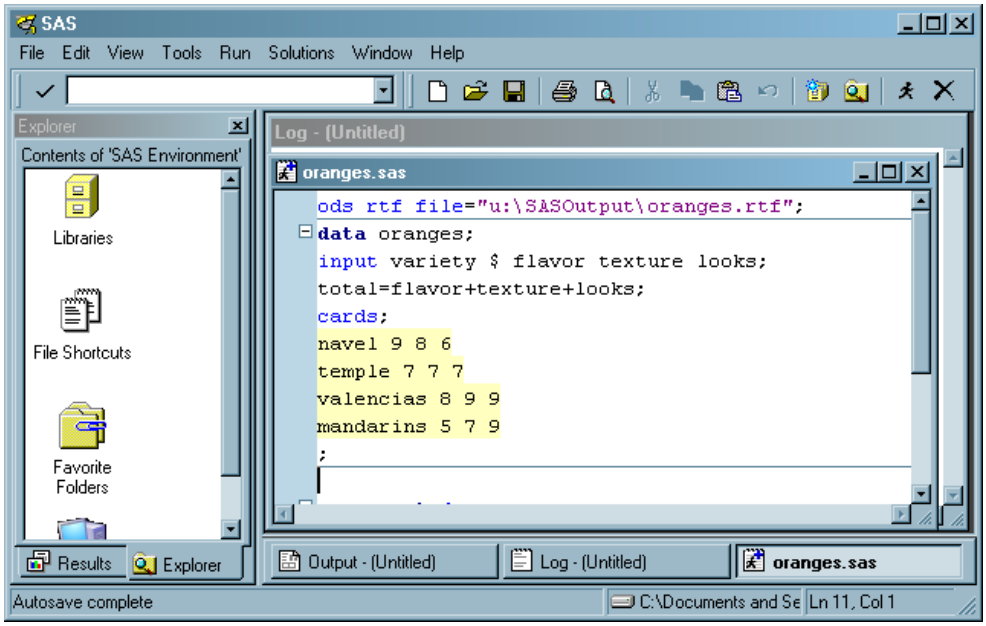
- invoking SAS from the Start menu
- submitting a sample SAS program

- examining the program output
- ending the SAS session.

You can invoke SAS from the Start menu by, selecting **Programs** ⇨ **SAS** ⇨ **9.4**.

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.

Figure 1.17 *Submitting the Sample SAS Program*

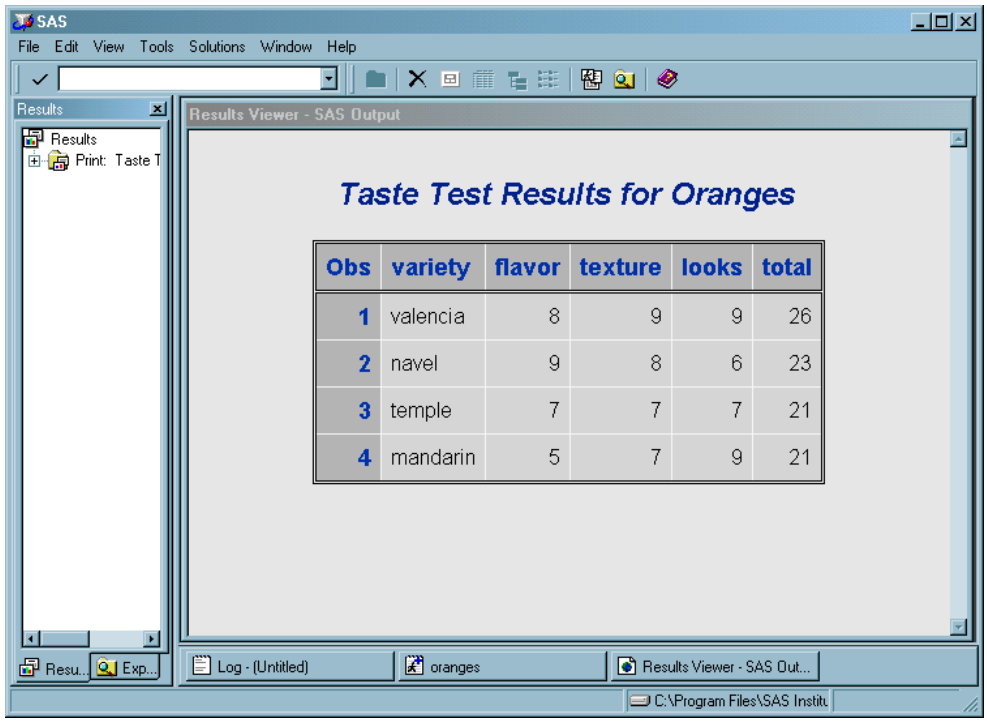


The following SAS code appears in the Enhanced Editor window:

```
ods rtf file="c:\em\oranges.rtf";
data oranges;
   input variety $ flavor texture looks;
   total=flavor+texture+looks;
   datalines;
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;
```

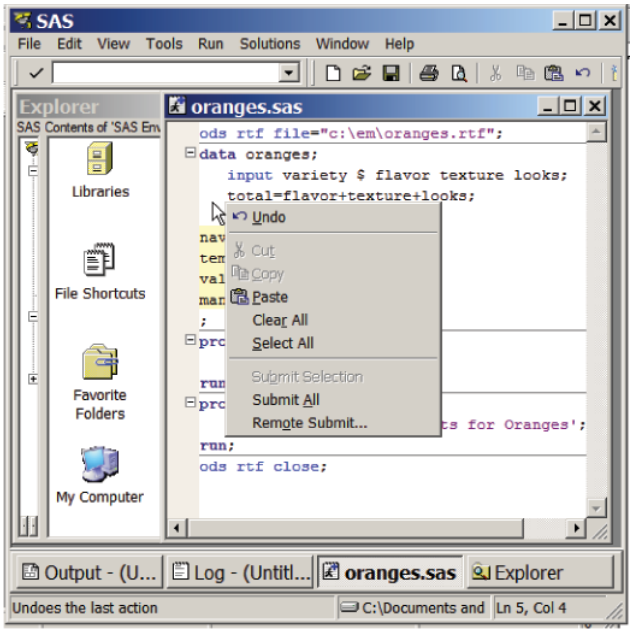
After you submit the program, the output appears in the Results Viewer window as follows:

Figure 1.18 *Looking at the Program Output*



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 right-clicking in an Enhanced Editor window.

Figure 1.19 *Pop-up Menu in the Enhanced Editor Window*



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 84 .

What If SAS Does Not Start?

If SAS does not start, the SAS log can 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, if it is available, or to the SAS console log, which is a Windows file. Under Windows the SAS console log is typically located in `c:\Users\`*user-ID*`\AppData`. You can obtain the location and 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**.

If SAS does not start, if the screen appears and then disappears, or if SAS is very slow to open, you might have a problem with a missing printer, a damaged printer driver, or a failed network printer connection. Use the following steps to correct this problem:

1. Verify that the printers are linking to valid network servers. If the printers are linking to invalid servers, then delete the printers by accessing **Start** ⇨ **Settings** ⇨ **Control Panel** ⇨ **Printers**.
2. Download a new printer driver from the printer's website and replace the current driver with the new driver.
3. Rename profile2.SAS7bcat to profile2.old and rename profile.SAS7bcat to profile.old at `c:\Users\`*user-ID*`\Documents\My SAS Files\9.4\`.
4. Start SAS.

Files Used by SAS

Introduction to Files Used by SAS

SAS uses many files while it is running. However, some of these files, such as the following, are especially important from a user's perspective.

- 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 more information about system options, see *SAS System Options: Reference*.

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.

In SAS 9.4, the location selected for SAS software installation is created and called SASHOME. !SASROOT is located at **!SASHOME/SASFoundation/9.4**. The default folder **!SASHOME** is located at **c:\program files\SASHome**. The default configuration location is **c:\program files\SASHome\SASFoundation\9.4\nls\en\sasv9.cfg**.

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\SASHOME\SASFoundation\9.4\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 **!SASHOME** folder, see "SAS Default Folder Structure" on page 40 .

Specifying System Options in a Configuration File

Any system option can be specified when you start SAS. It is often more convenient to place frequently used system options in a configuration file. For details about the syntax for specifying system options in a SAS configuration file, see "Syntax for System Options in the SAS Invocation or SAS Configuration File" on page 575 .

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 30 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.

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, thus 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.

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 are not specified. Under Windows, the path for the user profile folder is **c:\Users***user-ID***\Documents**

Note: If you create a configuration file in your home directory and the NOUSERCONFIG= system option is set, the configuration file in your home directory is skipped. For more information, see USERCONFIG System Option.

Starting SAS with an Alternate Configuration File

When you use a file that is located in a different folder or that has 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\SASHome\SASFoundation\9.4\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 34 and "CONFIG System Option: Windows" on page 605 .

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.

SASVx.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.

.SASVx.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, the user profile folder is **`c:\Users\`***user-ID*`\Documents\`**.

SASVx.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, the user profile folder is **`c:\Users\`***user-ID*`\Documents\`**

SASVx.CFG in the current folder
: SAS looks for a file that is named SASVx.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 you

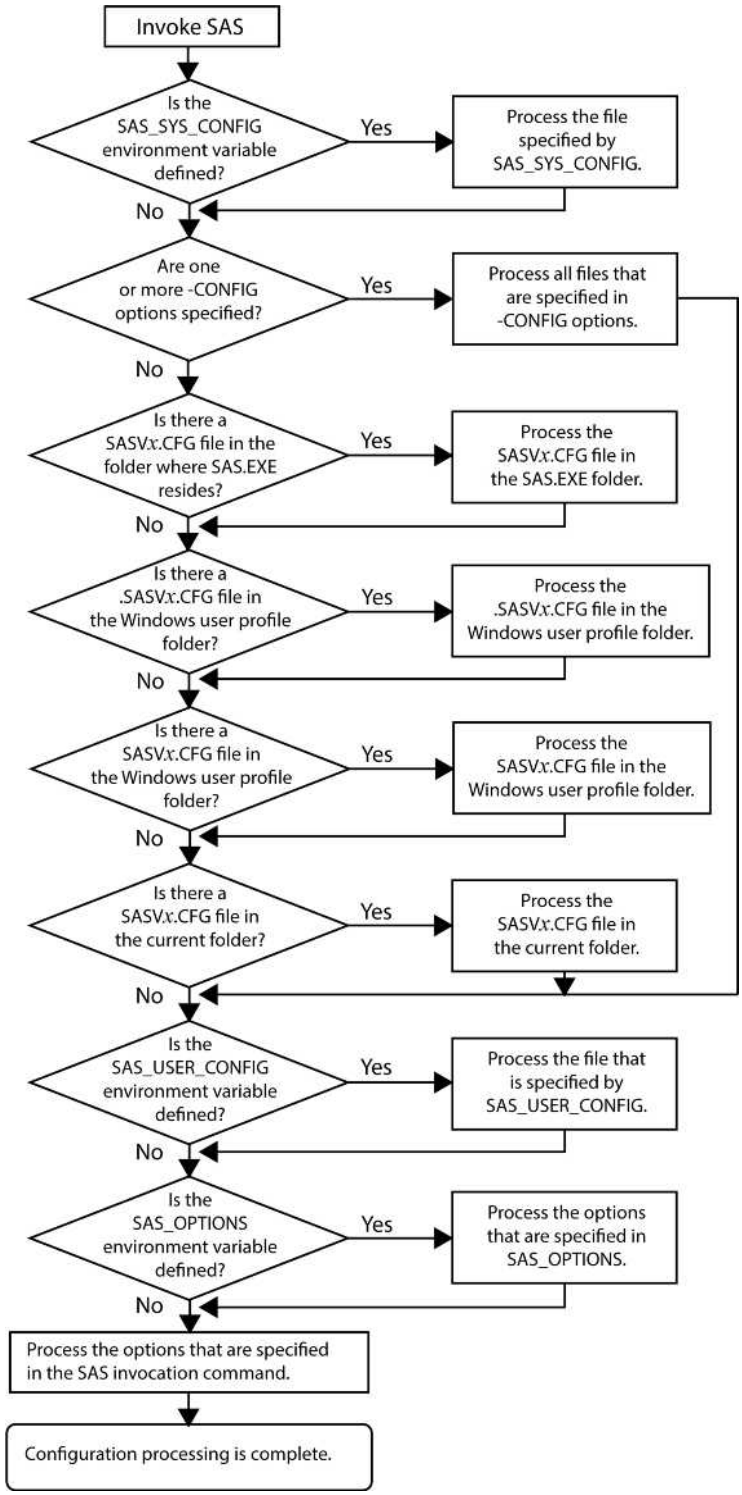
invoke SAS. For example, this environment variable might contain **-obs 2m -sasinitialfolder c:\myfolder -linesize max**.

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 Start menu, the last configuration file to be processed is the one that is specified in the Start menu shortcut for SAS. The options that are 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 1.20 *Order of Processing for SAS Configuration Files*



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 always override the option values that are specified in configuration files. Figure 1.20 on page 33 illustrates the flow of the SAS configuration file processing.

For more information about the CONFIG system option, see “CONFIG System Option: Windows” on page 605 .

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.

Note: For information about determining the current folder, see: "Determining the Current Folder When SAS Starts" on page 24 .

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: Windows" on page 593 .

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 you invoke SAS. You can 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\SASHome\SASFoundation\9.4\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 Options under Windows" on page 572 . For more information about the INITCMD system option, see *SAS System Options: Reference*.

Suppressing the Autoexec File

If you have an AUTOEXEC.SAS file in your current folder, but you want to suppress it, specify the NOAUTOEXEC option in the SAS command, as in the following example:

```
c:\program files\SASHome\SASFoundation\9.4\sas.exe -noautoexec
```

Profile Catalog

Introduction to the Profile Catalog

Each time 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 1.1 *The Default SASUSER Locations for the Windows Operating Environment*

Windows 7 and 8, Windows Server 2008 and 2012
`-sasuser "c:\Users\`*`user-ID`*`\Documents\My SAS Files\9.4"`

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\9.4`**). 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 quotation marks (") 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: Windows” on page 680 .

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 170 .

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 169 .

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.

Distributing Work Directories

SAS can make the distribution of Work libraries dynamic by distributing Work libraries across several directories. This functionality eliminates the potential problem of filling up a single volume with all of the Work directories. The WORK system option requires an argument, which must specify a .txt file that contains a list of directories that SAS uses for allocating Work libraries. Individual Work libraries reside in a single directory. The WORK system option is used in the sasv9.cfg configuration file or on the command line. When the argument WORK is a list of directories in a file, specify the method for choosing which directory to use for WORK. If you specify METHOD=RANDOM, then SAS chooses a directory from

the list of available directories. If you specify METHOD=SPACE, then SAS chooses the directory that has the most available space.

For more information, see the "WORK System Option: Windows" on page 717 .

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*nnnnnnnnn*, where TD means temporary folder and *nnnnnnnnn* is the process ID and nodename is the name of the host of the machine that created the folder. 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 might need to delete the temporary files.

Deleting the Work Folder

If SAS terminates abnormally, determine whether 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 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 !SASHOME. SAS Foundation products are installed in a folder structure under SASHOME called SASROOT. If you use the default provided by SAS, the SASHOME folder is **`c:\Program Files\SASHOME\SASFoundation\9.4`**. (The examples in this document assume the !SASHOME folder is called **`c:\Program Files\SASHOME\SASFoundation\9.4`**.)

The 32-bit content on an X64 system is installed in **`c:\program files\SASHome\x86\SASFoundation\9.4`**

SAS creates a folder for shared components, such as the Enhanced Editor and images, that are used by other SAS products. For SAS 8, the default path for shared components is **`c:\Program Files\SAS Institute\Shared Files`**. For SAS 9 through 9.4, the default path for shared files, if shared components are installed, is **`c:\Program Files\SAS\Shared Files`**. If shared components are not installed, the default path is **`c:\Program Files\SAS\SharedFiles`**. There is no blank space in the spelling of SharedFiles.

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 on-site SAS support personnel.

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 entered into the Enhanced Editor or Program Editor window, you issue the SUBMIT command. SAS provides several ways to do this:

- Press F3 or F8 when the editor window is active.
- Click the **Submit** toolbar button.
- Enter `submit` in the command bar.
- Select **Run** ⇨ **Submit**.

You can use the SUBTOP command from either the command line or the Run menu to submit one or more lines of your SAS code. For more information, see "SUBTOP Command: Windows" on page 414 .

Note: If you insert tabs while entering data in the DATALINES statement, you might get unexpected results when using columnar input. This issue exists when you use the SAS Enhanced Editor or SAS Program Editor. To avoid the issue, do one of the following:

- Replace all tabs in the data with single spaces.
- Use the %INCLUDE statement from the SAS editor to submit your code.
- If you are using the SAS Enhanced Editor, select **Tools** ⇨ **Options** ⇨ **Enhanced Editor** to change the tab size from 4 to 1.

Submitting Code from the SAS NOTEPAD Text Editor

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

From the Run menu in the Enhanced Editor or from the Program Editor window, select **Submit clipboard**. The code is submitted from the clipboard directly to SAS (without appearing in the Enhanced Editor or in the Program Editor window).Notes and results are sent to the SAS Log and to the Output window, respectively. The notes and results are displayed in the Results Viewer if it is active. You can still issue the RECALL command (or press F4) to recall the submitted program into the Enhanced Editor or 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: Windows” on page 407 .

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 onto 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 use one of the submit processes. See "Submitting Code from the Clipboard" on page 42.

If the application supports nondefault dragging of text, you can right-click the text to select it and then drop the text in the editor. When you drop the selection onto 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 onto the Enhanced Editor window or on the Program Editor window, that code is included in the window (but not submitted). If you drop the file onto 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 taskbar. You cannot drop a file onto the taskbar. Instead, you can drag the file to the SAS icon on the taskbar 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 onto 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 a time 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.

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 on what tasks are in progress when you interrupt your session, SAS can 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 Taskbar (or **End Task** from within the Windows Task Manager). If you use either of these methods, SAS displays a dialog box to enable 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.

Note: The X statement and X command are not valid in SAS Viya.

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: Windows” on page 560 .

These language elements can be used to run code using Windows or MS-DOS commands:

- X statement
- X command
- CALLSYSTEM routine
- %SYSEXEC statement.

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 that you define are not available to SAS. If you submit an X statement or command without an argument *command*, 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: Windows" on page 464 and the section on the CALL SYSTEM routine in *SAS System Options: Reference*.

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 an 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 **Command Prompt** on the Windows Taskbar. 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 that 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 that the command prompt session waits for you to type EXIT before you can return to your SAS session.

XSYNC
: means that 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 1.2 on page 49 .

Table 1.2 *Combining the XWAIT and XSYNC System Options*

Options in Effect	Result
XWAIT XSYNC	The command prompt window waits for you to type EXIT before closing, and SAS waits for the application to finish.
XWAIT NOXSYNC	The command prompt window waits for you to type EXIT before closing, and SAS does not wait for the application to finish.
NOXWAIT XSYNC	The command prompt window closes automatically when the application finishes, and SAS waits for the application to finish.
NOXWAIT 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 50 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:

- At a 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 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, the SAS console log is typically located in `c:\Users\`*user ID*`\AppData`. You can obtain the location and 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 under Windows or newer operating environments, expand the **Windows Logs** category and click **Application.**

2

Interacting with SAS under Windows

Overview of the SAS Interface **54**
The SAS Windowing Environment 54
Understanding Components of the Main SAS Window 54
Getting Help for the Main SAS Window 56

Working within Your SAS Session **57**
Using the Docking View 57
Using the Window Bar 59
Using Menus 60
Changing the SAS Current Folder 60
Issuing SAS Commands 63
Sending Email Using SAS 65
Saving Windows to External Files 77
Clearing the Window and Filename 77
Defining Keys 78
Navigating with Microsoft IntelliMouse 79
Using the Clipboard 79
Creating Text Highlighting and Special Characters 81

Customizing Your SAS Session **82**
Overview of Customizing Your SAS Session 82
Selecting Fonts 83
Setting Session Preferences 84
Customizing Your Windowing Environment with Commands 89
Customizing Your Windowing Environment with System Options 91
Main SAS Window 92
Customizing the Toolbar 95

Accessing Online Help and Documentation **102**
Getting Help from the Command Bar 102
Getting Help in the Dialog Boxes 102
Getting Help for a SAS Product 102
Getting Help from the Help Menu 103
Viewing Output and Help in the SAS Remote Browser 104

Overview of the SAS Interface

The SAS Windowing Environment

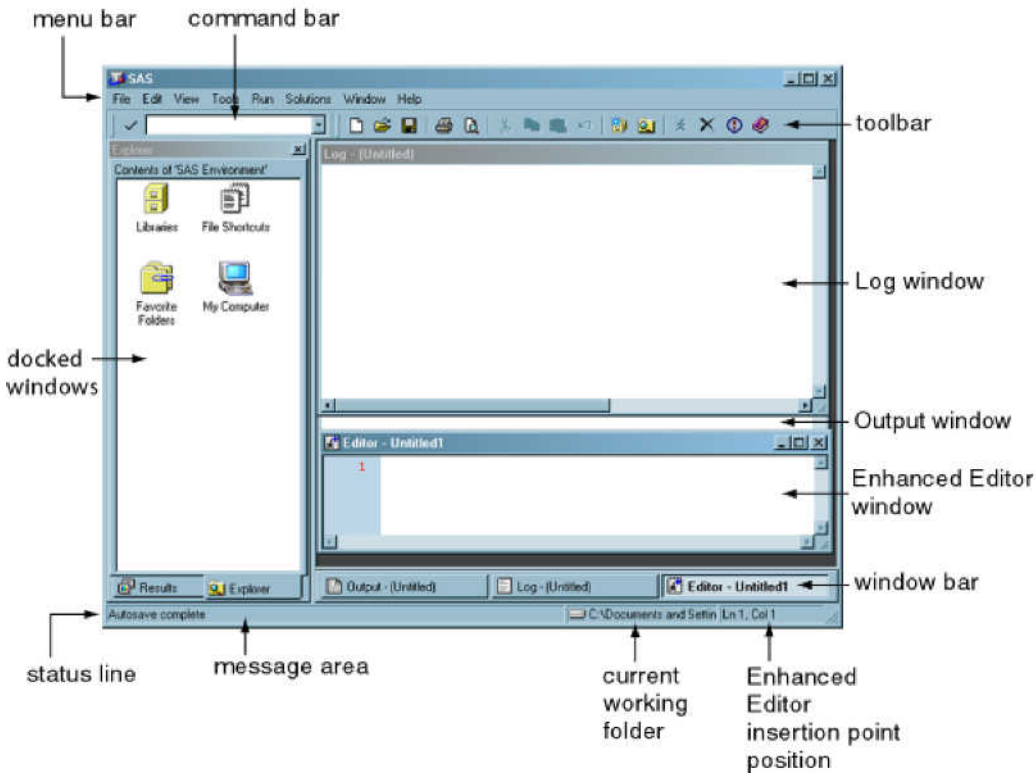
The SAS windowing environment refers to the windows that open in the main SAS window. You access the main SAS window when you start SAS from your PC or from a terminal emulator.

Windows in client software that do not access the main SAS window, such as Enterprise Guide, are documented in their product documentation. For more information about Enterprise Guide, see the Help in Enterprise Guide.

Understanding Components of the Main SAS Window

The main SAS window contains all other SAS application windows. The main SAS window is completely configurable, enabling 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*



Here 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 submits 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 that 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 bar, 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 you select **Tools** ⇨ **Options** ⇨ **Preferences**. Windows that cannot dock to the main SAS window appear 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 57 .

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 appears, a tab 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 by selecting **Tools** ⇨ **Options** ⇨ **Preferences View Tab** or by using the window bar pop-up menu.

status bar
: 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 60 .

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

  The status bar can be enabled and disabled by accessing **Tools** ⇨ **Options** ⇨ **Preferences**.

Getting Help for the Main SAS Window

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

For a description of a menu or a menu item:

1 Select the menu or menu item. A description of the item appears in the message area of the status bar.

2 For example, when you select the **File** menu, the message `Perform file-related operations` appears in the **Status Bar**.

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

To access help on other parts of the main SAS window:

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 bar help text for commands available from the toolbar see “Customizing the Toolbar” on page 95 .

To enable or disable command bar or toolbar ScreenTips, you can use the **Show ScreenTips on toolbars** option by selecting **Tools** ⇨ **Customize Toolbar Tab** or enter the TOOLTIPS 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** about enabling and disabling ScreenTips, see “Setting Session Preferences ” on page 84 , “Customizing the Toolbar” on page 95 , “WSCREENTIPS Command: Windows” on page 439 , and “TOOLTIPS Command: Windows” on page 419 .

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 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 86 . To use a command to dock and undock the docking view, see "WDOCKVIEW Command: Windows" on page 424 .

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 "WDOCKVIEWRESIZE Command: Windows" on page 426 .

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: Windows" on page 425 and "WDOCKVIEWRESTORE Command: Windows" on page 426 .

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 bar pop-up menu
- the window bar pop-up menu
- entering `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 entering `wscreentips` on the command bar. For more information about enabling ScreenTips, see "View Preferences" on page 86 and "WSCREENTIPS Command: Windows" on page 439 .

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 load a file:

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 window bar 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.

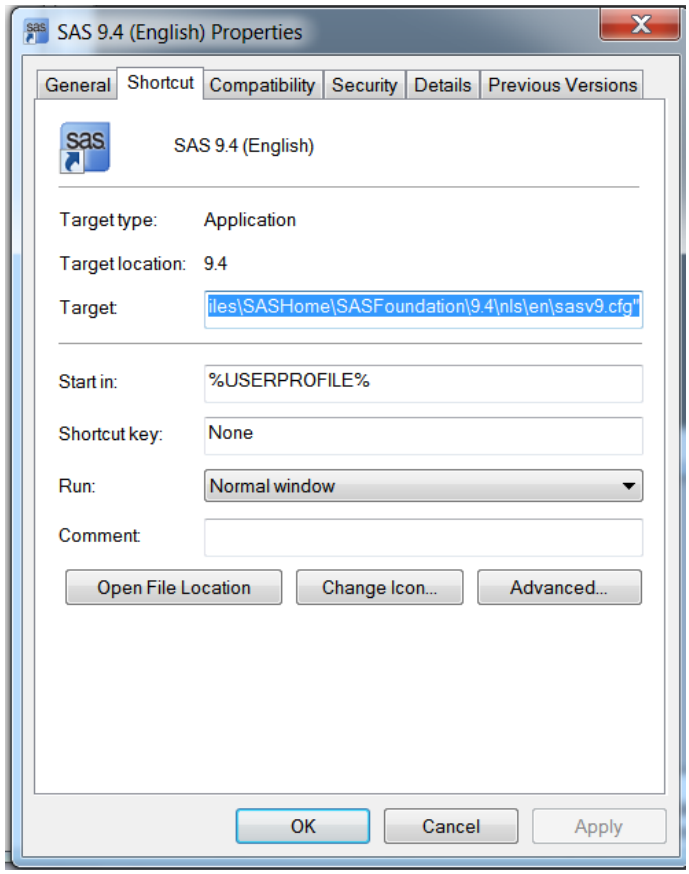
Some SAS windows (such as the Explorer window) along with the main SAS window can contain objects that have their own pop-up menus when you right-click an object. For example, the command bar, the toolbar, and the status bar each have a pop-up menu. In these windows, the pop-up menu is specific to the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

Changing the SAS Current Folder

What Is the Current Folder?

The current folder is the operating environment folder to which many but not all of the SAS commands and actions apply. The current folder is displayed in the status bar 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 on 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: Windows” on page 679 .

The following display shows the **Start in** field.



Interactively Selecting a New Current Folder

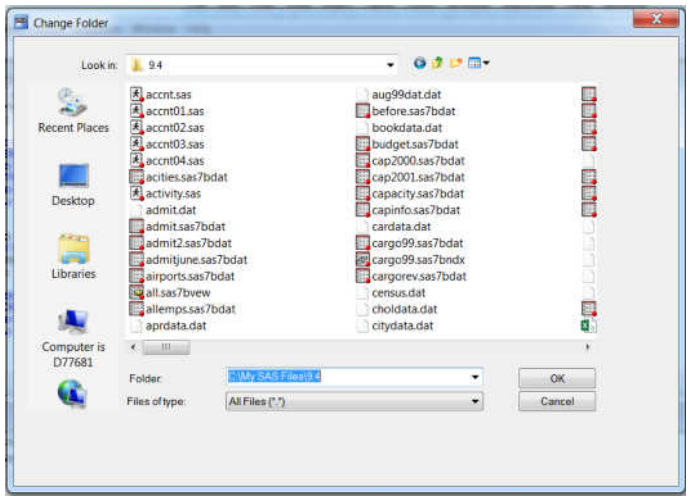
There are two methods to interactively select a new folder. Here is one method:

1 Select **Tools** ⇨ **Options** ⇨ **Change Current Folder**

2 Select a folder from the **Change Folder** dialog box.

Another method to change the SAS current folder during your SAS session is to double-click the current folder in the status bar. Then use the **Change Folder** dialog box (shown in the following display) to select a new current folder.

Figure 2.2 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 the 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** menu and then selecting **Submit** is the same as entering `submit` in the command bar.

The items in the menu bar and pop-up menu vary depending on the 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** to clear the option.

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 95 .

Using the Command Bar to Issue Commands

The command bar, as shown in Figure 2.1 on page 55 , is an integrated command line that offers a central location from which you can enter any SAS command that is valid for the active window. The command bar can also be undocked and can appear in a separate window that can be moved. It can be moved anywhere on your desktop. If you enter a command that is not valid for the active window, an error message is displayed on the status bar.

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**.

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. The following steps are the default method. 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: Windows” on page 386 . For more information about the **Customize Tools Toolbars** tab, see “Setting General Toolbar Preferences” on page 95 .

Using the Command Line to Issue Commands

You can activate 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, using the INCLUDE command on the command line of the Enhanced Editor window applies only to that window.

To activate 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 84 and “COMMAND Command: Windows” on page 386 .

Sending Email Using SAS

Overview of Sending Email

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)
- VIM (Vendor Independent Mail)
- SMTP (Simple Mail Transfer Protocol).

You might need to install an email client that supports one of these protocols before you can use SAS email support. Also, although you can use SAS to send messages, you must use your email 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 email. 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 email. If you specify SMTP as the email system in the EMAILSYS system option and you are using an email dialog box, the MAPI email system is used. For information about using SMTP, see *SAS Programmer's Guide: Essentials* as well as "FILENAME Statement: EMAIL (SMTP) Access Method" in *SAS Global Statements: Reference* , the "EMAILHOST= System Option" in *SAS System Options: Reference* , and the "EMAILPORT System Option" in *SAS System Options: Reference* .

Initializing Email

To send email from within SAS, use the following system options that are appropriate for your email system in the SAS configuration file or when you invoke your SAS session. These email options can be set anytime in the configuration file, when invoking SAS, or during the SAS session.

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

Note: The directory that contains the email DLL file (for example, MAPI32.DLL or VIM32.DLL) must be specified in your Windows PATH environment variable. SAS uses the first email DLL that it finds for the interface that you specify.

-EMAILDLG NATIVE | SAS
specifies whether to use the native email interactive dialog box that is provided by your email application or the email 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 email access for your site. This option is necessary only if you are using the SMTP email interface. Multiple SMTP servers can be specified:`EMAILHOST SMTP server| ('server1' 'server2' <...'server-n'>)`

For more information about the EMAILHOST option and more suboptions, see the EMAILHOST option in the *SAS System Options: Reference*

-EMAILAUTHPROTOCOL *authorization protocol*
specifies the authorization protocol to use in SMTP email. The default for this option is NONE. The only supported protocols are LOGIN and PLAIN.

-EMAILID *VIM email login ID* | *MAPI profile* | *email address*
specifies either your VIM email login ID, the MAPI profile that you use to access the underlying email system, or a fully qualified email 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 email interface. The default port is 25.

-EMAILPW "*password*"
specifies your email 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. Passwords can be encoded. You can create an encoded password with PROC PWENCODE.

Using Your Email Software to Send Mail

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

- Select **File** ⇨ **Send Mail**.
- Type DLGSMAIL in the command bar.

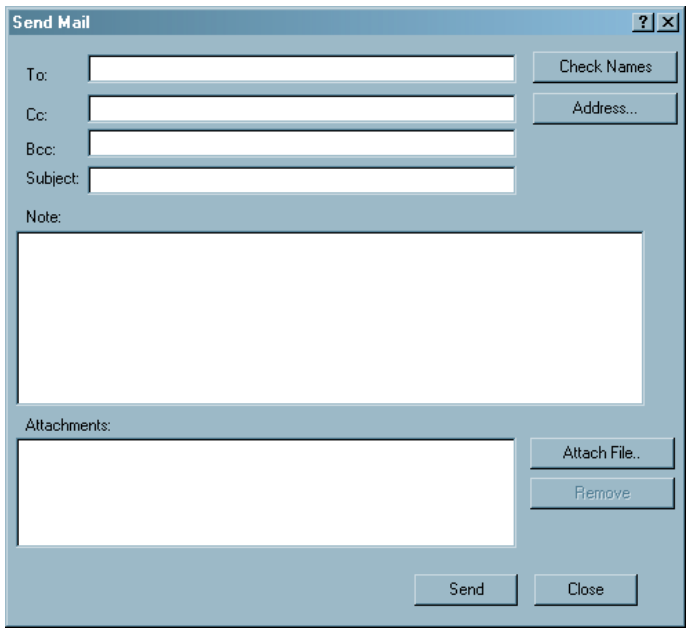
Your email software provides the interface to send mail.

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 email system by using the **Send Mail** dialog box, select **File** ⇨ **Send Mail**.

Figure 2.3 *Send Mail Dialog Box*



The **Send Mail** dialog box contains the following fields:

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

Cc
the email 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 receives a copy of the email. These addresses are not visible to those individuals in the TO and CC options.

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

Note
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 email systems currently limit the note length to 32K (or 32,768 characters). Text that you type in the **Note** area are automatically wrapped 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 email 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 email program that you are using and on the configuration of your network. For example, suppose you want to send email 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 email program, provided that the facility is accessible.

Sending the Contents of a Window by Email

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

Table 2.1 *File Type for Email Messages (by Active Window)*

File Type	Active Window
.txt	Log Output Enhanced Editor Program Editor
.bmp	Explorer Graphics 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 Email

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

- You can use the logic of the DATA step or SCL to subset email distribution based on a large data set of email addresses.
- You can automatically send email upon completion of a SAS program that you submitted for batch processing.
- You can direct output through email based on the results of processing.
- You can send email 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 email recipients, subject, and any attached files
- PUT statements that contain the body of the message
- PUT statements that contain special email directives (of the form !EM_*directive*!) that can override the email attributes (TO, CC, BCC, SUBJECT, ATTACH) or perform actions (such as SEND, ABORT, and NEWMSG).

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

The FILENAME statement syntax to send email 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 email.

'address'
is the valid destination email 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= email option or the PUT statement !EM_TO! directive, which overrides an address specification.

email-options
can be any of the following:

Note: Each email option can be specified only in a FILENAME statement that overrides the corresponding SAS system option.

EMAILID= *email login ID* | *MAPI profile* | *email address*
specifies your email login ID, MAPI profile name, or your SMTP email address that is used to access the underlying email 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 email option can be specified in the FILENAME statement that overrides the SAS system option.

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

EMAILSYS=MAPI | VIM | SMTP
SAS supports three types of email 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.

SMTP Simple Mail Transfer Protocol.

TO=*to-address*
specifies the primary recipients of the email. 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 receives a copy of the email. 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 receive a copy of the email. These addresses are not 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 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

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.

V | D indicates variable format. This value is the default.

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 email system is SMTP, see "FILENAME Statement: EMAIL (SMTP) Access Method" in *SAS Global Statements: Reference* 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 email fileref as the output destination, use PUT statements to define the body of the message. For an example of using email options in the FILE statement, see Example Code 2.2 on page 73.

You can also use PUT statements to specify email 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. Here 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 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!
abends 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 Email 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 email 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!';
    datalines;
Susan          marketing
Jim            marketing
Rita           development
Herb           development
;
run;
```

The resulting email 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 email options in the FILENAME statement instead of in the FILE statement. This method overrides the values for the SAS system options EMAILID, EMAILPW, and EMAILSYS.

```
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 Email Using SCL Code

The following example is the SCL code that generates a FRAME entry that is designed for email.

The FRAME entry has objects that enable the user to enter 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 2.4 *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 Email Using SMTP

The following examples show how you can send email 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 email.

To use SMTP that you need an SMTP email 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 email:

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

You can also send an attachment by using the ATTACH email option in the FILENAME statement. Compress non-textual attachments such as SAS data sets, bitmap files, and HTML files before using the ATTACH email 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 or select the **File** 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** 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. You can also select a file type from the **Save as type** list. SAS saves most file types as plain text and assigns different file extensions based on the type that 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 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
- Select the **New** (the blank page) button

- 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**. You can also press F9. These key definitions apply to 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 767 and "Keyboard Shortcuts within the Enhanced Editor" on page 772 .

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. Enter 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 enter 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 menus in the specific product window to access key definitions for specific products.

Navigating with Microsoft IntelliMouse

SAS provides support for mouse devices like 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 scroll bar 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 scroll bar.

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

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 that 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** 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** 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 that you are marking is in bitmap format. After you finish marking an area, you can copy it to the clipboard. If the window that 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, enable you to paste bitmaps into the window. For more information, see “Pasting an OLE Object from the Clipboard” in the “Using OLE in SAS/AF Software” chapter of the *SAS Companion for Windows*.

Also, you can paste bitmaps into the SAS GRAPH window to import graphics. For more information, see “Importing Graphics from Other Applications” on page 237 .

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 feature 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** 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 is 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 can define an icon for the command in the toolbar, or assign the GSUBMIT command to a function key. For more information about how to define buttons, see "Customizing the Toolbar" on page 95 .

If you submit SAS code from the Windows clipboard while a procedure RUN group is active, the submit fails. 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 81 and Table 2.3 on page 82 . You can enter 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

Overview of Customizing Your SAS Session

You can customize your SAS session by selecting fonts, setting session preferences, customizing your windowing environment, customizing the toolbar and other customizations.

The following sections in the *Step-by-Step Programming with Base SAS* provide information about customizing SAS sessions including saving settings between sessions:

- "Customizing Your Current Session " in *Step-by-Step Programming with Base SAS*
- "Customizing Session-to-Session Settings" in *Step-by-Step Programming with Base SAS*

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. The Enhanced Editor does not require mono space font.

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 is 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 might 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" on page 84
- "General Preferences" on page 85
- "View Preferences" on page 86
- "Edit Preferences" on page 86
- "Results Preferences" on page 87
- "Web Preferences" on page 88
- "Advanced Preferences" on page 88

Using the Preferences Dialog Box

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

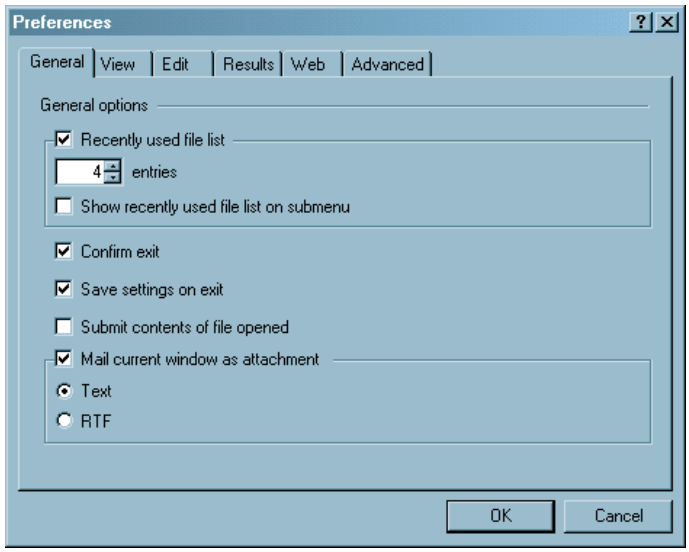
- Type `dlgpref` in the command bar
- Select **Tools** ⇨ **Options** ⇨ **Preferences**

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.

You must save the settings when you exit, do not have RSASUSER set, and are using sasuser.profile.

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



General Preferences

The **General** tab enables you specify the general options that control how your SAS session works. Here 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 the number of files, up to 30, that you want to retain. **Show recently used file list on submenu** specifies whether the files are 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 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 email 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 bar
specifies which aspects of the status bar, if any, you want to have visible in your session. **Display message lines** specifies whether to display the message area of the status bar. **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 enter 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, the pathname for the Application Data folder is `c:\Users\` *user ID*`\AppData`. For more information, see Autosave

The Program Editor contents are saved to **pgm.asv** in the current active folder so that you can recover your work if your SAS session ends before you 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.

Safe save (Enhanced Editor only)
enables the **Safe save** option, which creates a temporary file and then renames it to the original filename. If the file that is being saved resides in a WebDAV or SharePoint location, it is overwritten, and safe save is not needed.

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.

Note: If you want to display your output using the Listing format, then you must select the Listing option from the **Preferences** window.

HTML

Create HTML
by default, specifies to display program output in HTML format.

Folder
specifies a folder to store HTML output files. You can either enter 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 "Overview" in *SAS Output Delivery System: Procedures Guide* .

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

View results using
specifies supported browsers: Internal Browser, Internet Explorer, Chrome, and Firefox.

Use ODS Graphics
specifies to automatically generate graphs when running procedures that support ODS graphics.

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: Windows" on page 422 . 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 enter 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 access when invoking the web browser within SAS. By default, the browser navigates to https://www.sas.com/en_us/home.html

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 12.

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

When **Scroll max** is selected, no output is 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 runs.

You can also set these values by using the Editor Options window or the AUTOSCROLL command. For more information, see "AUTOSCROLL Command: Windows" on page 382 and the SAS Help and Documentation.

Other

The following settings are miscellaneous options settings:

Hide cursor in non-input windows

specifies that the insertion point does 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 setting 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 55) 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**

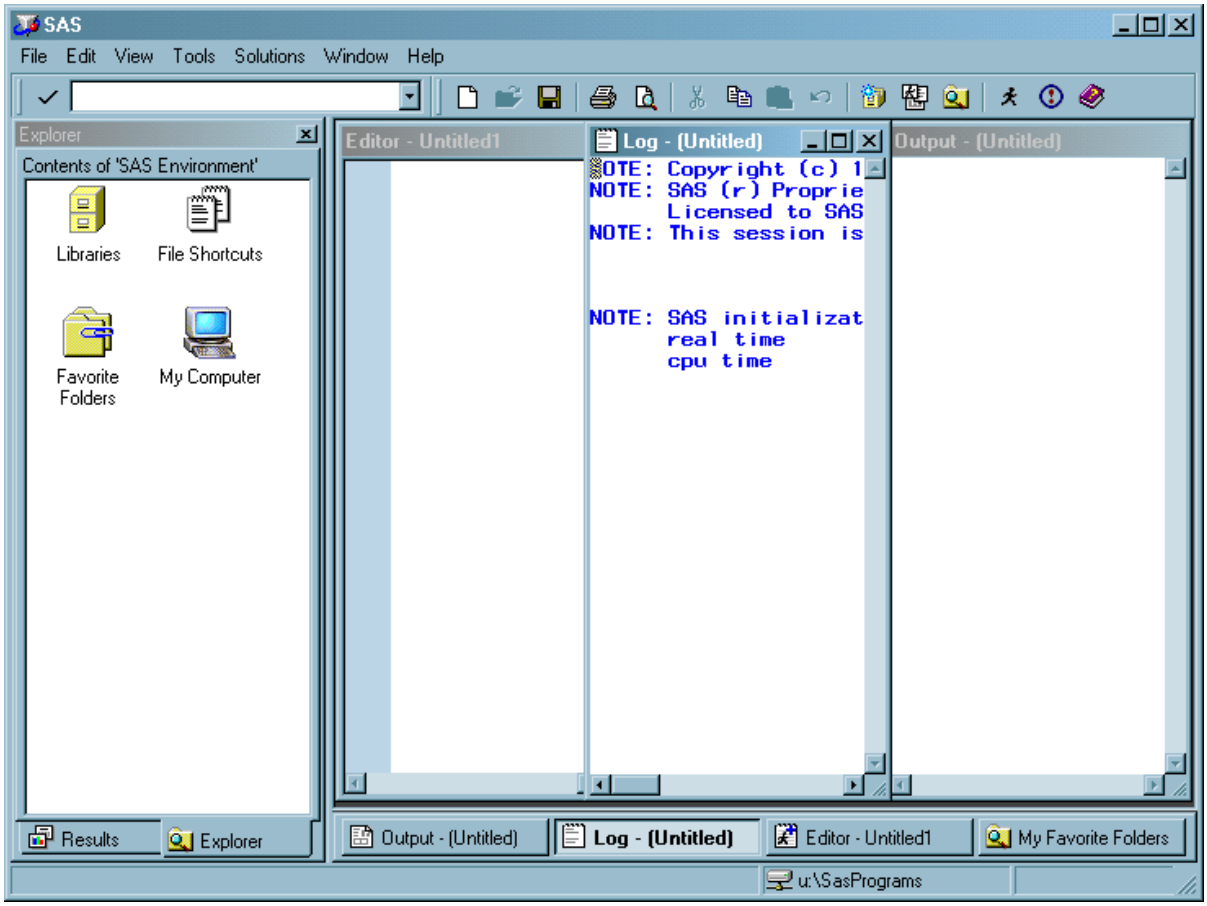
You can open more windows for easy access and rearrange the windows within the main SAS window. For example, you can keep the My Favorite Folders window open, but minimized, and the windows arranged in a mosaic pattern so that you can see all of them at once.

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:

Figure 2.5 Customized SAS Session



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 57 .

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 365 .

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, open the SASCOLOR dialog box:

- 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. See "Changing the Size and Placement of the Main SAS Window" on page 92 .

AWSTITLE
: specifies the text for the main SAS window title bar. See "Changing the Title of Your SAS Session" on page 92 .

HELPREGISTER
: enables you to add Help to the main SAS window **Help** menu in order to access custom Help. See "HELPREGISTER System Option: Windows" on page 627 .

ICON
: minimizes the SAS window when SAS initializes. See "Minimizing Your SAS Session" on page 92 .

REGISTER
: enables you to add applications to the main SAS window Tools menu so that you can execute them by clicking their names. See "Adding Applications to the Tools Menu" on page 92 .

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. See "SASINITIALFOLDER System Option: Windows" on page 679 .

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. See "Displaying a Custom Logo Screen during SAS Invocation" on page 93 .

USERICON
: specifies user-defined icons to be incorporated into SAS/AF applications. See "Adding User-Defined Icons to SAS" on page 94 .

WEBUI
: specifies to enable use of the pointer to select an object and a single click to invoke the object's default action. See "Enabling Web Enhancements in SAS" on page 94 .

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 Options under Windows" on page 572 . For a comprehensive list of these options, see "SAS Options That Control the Main SAS Window" on page 364 .

Main SAS Window

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: Windows" on page 596 .

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: Windows" on page 598 .

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: Windows" on page 632 .

Adding Applications to the Tools Menu

The REGISTER system option enables you to add names of applications to the Tools 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.

Here 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 menu, the output of the Windows DIR command is displayed in a command prompt window.

Here 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 action 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: Windows" on page 670 .

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 SASINITIALFOLDER 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: Windows" on page 679 .

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: Windows" on page 694 .

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: Windows" on page 712 and refer to the SAS Help and Documentation for SAS/AF software and SAS/EIS software.

Enabling Web Enhancements in SAS

If Microsoft Internet Explorer is 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 can customize the toolbar settings to access commonly used commands or create a toolbar with a specific application window. This section describes how to customize the toolbar settings.

- "Using the Customize Tools Dialog Box" on page 95
- "Setting General Toolbar Preferences" on page 95
- "Customizing a Toolbar" on page 96
- "Adding a Tool to the Toolbar" on page 98
- "Removing a Tool from the Toolbar" on page 99
- "Customizing and Saving a Toolbar for Use with a Particular Application or Window" on page 99
- "Resetting the Tools to the Default Settings" on page 100
- "Examples of Useful Tools That You Can Create" on page 100

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 setting 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.

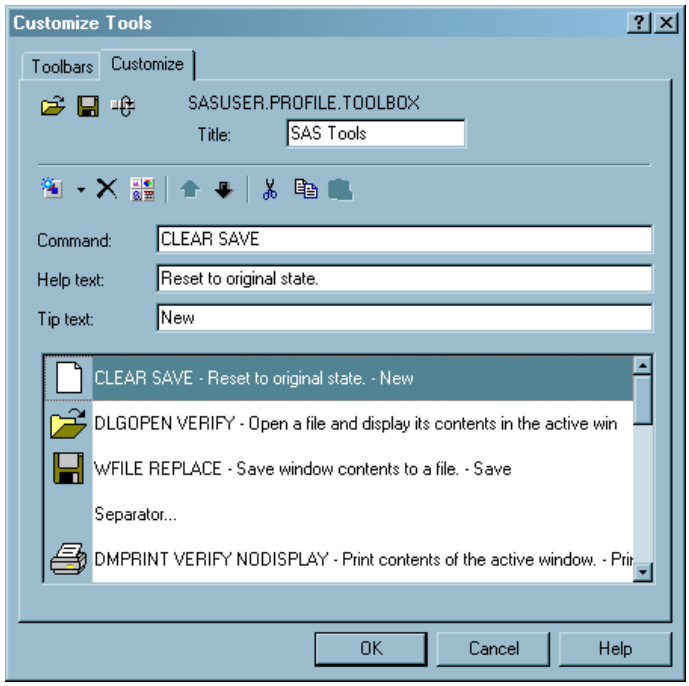
- When **Use autocomplete** is selected, SAS stores commands that were entered previously and completes the command once you start entering the command.
- Select **Sort commands by most recently used** to display commands in the command bar list by the most recently entered command. If this setting is not selected, the commands are ordered by the most frequently used.
- In **Number of commands saved**, enter 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.

Figure 2.6 Customize Tab of the Customize Tools Dialog Box



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 that 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 add 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.

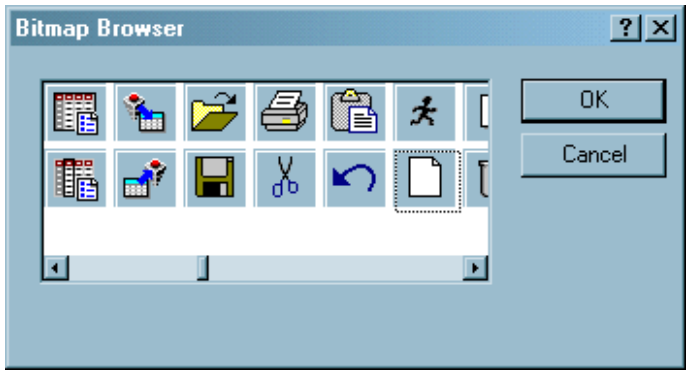
Toolbar list
: lists the buttons, commands, Help text, and separators that are defined in the toolbar.

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 selection 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, enter the library, catalog, and toolbox name. Then click **OK**.

Figure 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 menu. Buttons are enabled only if the command is available from a 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.

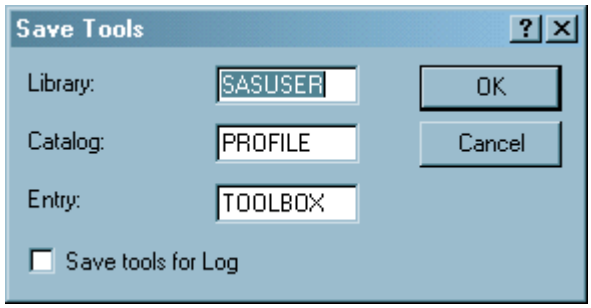
3 When you are finished customizing the toolbar, click **Save**. The **Save Tools** dialog box appears (as shown in Figure 2.8 on page 100).

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.

Figure 2.8 Save Tools Dialog Box



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: Windows" on page 417 .

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 That 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 action 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 website 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:

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

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

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

WPGM; 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*.

WPGM; 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

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, and then click the item that 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 does not 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.

- 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

Overview of 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. Selecting this topic 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 helps 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 website, including Technical Support, frequently asked questions, sending feedback to SAS, and the SAS home page.

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 Websites) from within SAS

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

- Enter a URL (uniform resource locator) in the command bar. SAS launches the browser that you specified in the **Web** dialog box under **Preferences** .
- Type `wbrowse` in the command bar. This action 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: Windows" on page 422 .

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

Viewing Output and Help in the SAS Remote Browser

What Is Remote Browsing?

You can use remote browsing to view SAS Help and Documentation and the following types of documents in the web browser on your local computer:

- URLs that are specified in the WBROWSE command
- ODS output

By displaying ODS output locally with remote browsing, you have access to output that requires browser plug-ins that are not available for Windows 64-bit servers.

A software agent that is called the remote browser server runs on your local computer. When SAS needs to display content, SAS connects to the remote browser server and sends to the remote browser server the URL that references the content. The remote browser server then passes the URL to your browser for display. If the remote browser server is not running on your computer, SAS displays the URL that you need to download to the remote browser server.

Remote browsing is supported on Windows 64-bit servers. The remote browser supports PDF, RTF, XLS, and other types of output.

Remote Browsing and Firewalls for General Users

If your network has a firewall between desktop computers and the computer that is hosting SAS, web browsers cannot display web pages from your SAS session. Usually, this problem is indicated by a time-out or connection error from the web browser. If you receive a time-out or connection error, contact your system administrator.

Remote Browsing and Firewalls for System Administrators

To enable the display of web pages when a firewall exists between desktop computers and the computer that is hosting SAS, a firewall rule must be added, so the web browser can connect to SAS. The firewall rule specifies a range of network ports for which SAS remote browsing connections are allowed. Contact the appropriate system administrator who can select and configure a range of network ports for remote browsing. The range depends on the number of simultaneous SAS users. A value of approximately three times the number of simultaneous SAS users should reserve a sufficient number of network ports.

After the firewall rule is added, SAS must be configured to listen for network connections in the network port range. Normally, SAS selects any free network port, but the HTTPSERVERPORTMIN and HTTPSERVERPORTMAX system options limit the network ports that SAS selects. To add these system options to your SAS configuration file:

1. Set HTTPSERVERPORTMIN to the lowest port in the network range.
2. Set HTTPSERVERPORTMAX to the highest port in the network range.

For example, if the system administrator defined a network port range of 8000 to 8200, the system options would be set as follows:

```
httpserverportmin=8000
httpserverportmax=8200
```

After these system options are set, desktop computers can display web pages. If there is an insufficient number of ports or if the system options are specified incorrectly, a message appears in the SAS log.

For more information about these options, see "HTTPSERVERPORTMIN= System Option" in *SAS System Options: Reference* and "HTTPSERVERPORTMAX= System Option" in *SAS System Options: Reference*.

Using Remote Browsing with ODS Output

The SAS Output Delivery System (ODS) can be used to generate graphical reports of your SAS data. Remote browsing enables you to view your output directly from a SAS session either in real time as the output is generated, or on demand from the Results window.

Remote browsing displays ODS output in PDF and RTF formats. If your browser does not have the appropriate plug-in for non-HTML data types, the browser displays a dialog box rather than the output. This dialog box enables you to download the report to your PC and view it using a local program, such as Excel for an XLS file.

The automatic display of ODS output (PDF and RTF formats) is turned off by default. You can turn on the automatic display of ODS output by issuing the AUTONAVIGATE command in the Results window.

Installing the Remote Browser Server

You can install the remote browser server directly from your SAS session. If SAS is unable to make a connection for remote browsing, SAS displays a dialog box that contains the URL that you need to download the installer. Use this URL to download and install the remote browser server. Do not exit SAS. To install the remote browser server:

1. Type the URL that appears in the dialog box into your web browser and press Enter.
2. After the download page is displayed, download the installer that is appropriate for your computer.
3. Run the installer.

Note: In the Windows environment, the remote browser server is added to your Start-up items, so that the server starts whenever you log on. An icon is displayed in your system tray to indicate that the remote browser server is running.

System Options for Remote Browsing

After the remote browser server is running on your computer, you can run remote browsing by specifying the HELPHOST and HELPPORT system options.

- The HELPHOST system option specifies the name of your host computer. If you do not specify this option, then the host name that is specified in the X Windows display is used. For more information, see "HELPHOST System Option: Windows" on page 623 .
- The HELPPORT system option specifies the port number for the remote browser server that is installed on your computer. For more information, see the HELPPORT system option in the *SAS System Options: Reference*.

You can set these options in your configuration file, at SAS invocation, or during your SAS session in either the OPTIONS statement or in the SAS System Options window.

3

Using the SAS Editors under Windows

Using the Enhanced Editor **108**
Enhanced Editor Features 108
Using the Enhanced Editor Window 108
Creating Your Own Keywords 131
Associating File Extensions with File Types 132
Setting Enhanced Editor Options 133
Using Keyboard Shortcuts to Customize the Enhanced Editor 138
Accessing the Enhanced Editor 140

Using the Program Editor **140**
Overview of Using the Program Editor 140
Switching from the Enhanced Editor to the Program Editor 140
Opening Files 141
Using Line Numbers 142
Moving the Insertion Point 142
Using Tabs 143
Understanding Line Breaks 143
Selecting Text 143
Deleting Text 144
Finding and Replacing Text 144
Dragging and Dropping Text 146
Drag Scrolling 147
Using Rich Text Format Text 148
Saving Files 148
Saving Program Editor Files Using Autosave 148
Understanding Unique Features of the Editor 149

Using the Enhanced Editor

Enhanced Editor Features

While retaining some familiar Program Editor features, the Enhanced Editor enables you to do the following:

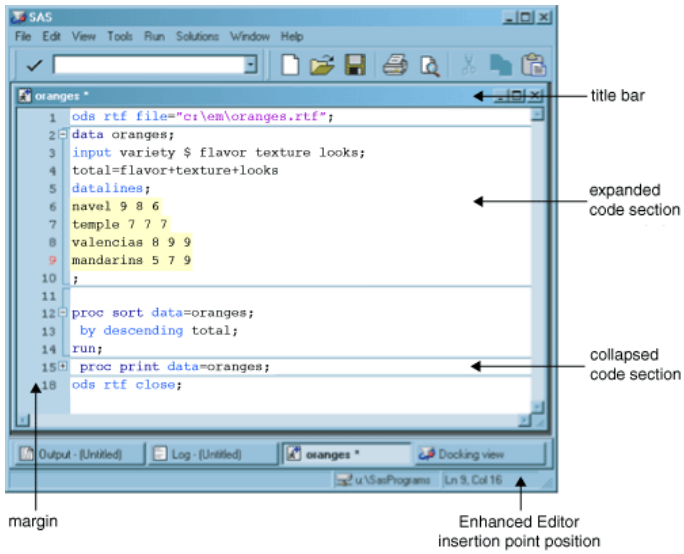
- use color-coding to identify SAS and SCL program elements as well as HTML and XML document elements. Color-coding settings can be saved in a color scheme.
- create and format your own keywords.
- automatically indent the next line when you press Enter.
- view the high-level flow of your SAS program 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 entering 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 file. You can open multiple files (.sas program files) in multiple Enhanced Editor windows.
- 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:

Figure 3.1 *Enhanced Editor Window*



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 do the following:

  - 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 taking either of the following actions:

- selecting an Enhanced Editor window
- entering `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*

Actions	Instructions
Open a new file	Do one of the following: ■ Type **`wedit`** in the command bar ■ Select **View** ⇨ **Enhanced Editor** ■ Click the **New Program** toolbar button ■ Select **File** ⇨ **New Program**
Open an existing file by using the Open dialog box	Open the dialog box by using one of the following: ■ Click the **Open Program** toolbar button ■ Select **File** ⇨ **Open Program** ■ Type *fileopen* in the command bar.* Select the file. Click **Open**. **Note:** You can also open an existing file by using the **Most Recently Used File List**. You can set the Recently Used File List option by accessing **Tools** ⇨ **Options** ⇨ **Preferences** ⇨ **General Tab**. When you set this option, files that are created or opened are listed at the bottom of the **File** menu.
Open multiple individual files by using the dialog box	1 Hold down the Ctrl key. 2 Select the files. 3 Click **Open** .
Open an existing file, bypassing the **Open** dialog box	Do one of the following: ■ Type **`wedit`** "*filename*" in the command bar. ■ Type **`fileopen`** "*filename*" in the command bar.*
Open multiple views of an opened file	1 Make the file the active window. 2 Select **Window** 3 Select **New Window**

Actions	Instructions
	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 to the active window. 2 Select **File** 3 Select **Append** 4 Select a folder and the file to append. 5 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: Windows" on page 679 and "Changing the SAS Current Folder" on page 60 .

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** 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 storage device). If the file is to be saved for the first time, the **Save As** dialog box appears 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 Enter a filename in the **Filename** field.

4 Select a file type from the **Save as type** field.

5 Click **Save**.

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: Windows" on page 679 and "Changing the SAS Current Folder" on page 60 .

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, the pathname for the Application Data folder is **`c:\Users\`***user-ID*`\AppData\Roaming`**. For example, the path to the autosave file for MYPROGRAM.SAS in folder C:\TEMP would be

```
C:\Users\my-user-ID\AppData\Roaming\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 can find the autosaved file under the original filename.

The application data folders are hidden by default by the operating system. To view the folders from the operating system:

1. Select **Start** ⇨ **Computer** ⇨ **Organize** ⇨ **Folder**.
2. Select **Search Options** ⇨ **View**.
3. Select **Show hidden files and folders**.
4. Deselect **Hide extensions for know file types**.
5. Select **Apply** and **OK**.

For information about setting the autosave feature, see "Edit Preferences" on page 86 and "WAUTOSAVE Command: Windows" on page 421.

Note: To prevent the loss of data when overwriting an existing file, the Enhanced Editor saves the contents to a temporary file, *SASuuuu.tmp*, where *uuuu* is a unique hexadecimal value. The original file is replaced with the temporary file and the original file's created date is preserved. After you save and replace the files, the temporary file should not exist. If the temporary file still exists, verify that the contents of the original file have been updated, and then delete the temporary file.

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. Here is an 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. All Enhanced Editor commands can be entered 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 entering `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. These commands are entered in the main command bar or in the command bar that is specific to that window. 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 Inserts 4 lines after the current line

Command	Description	Default Value of *n*	Maximum Value Allowed	Example
:IA*n*	Insert *n* lines after the current line.	1	9999	:IA4 Inserts 4 lines after the current line
:IB*n*	Insert *n* lines before the current line.	1	9999	:IB2 Inserts 2 lines before the current line
:D*n*	Delete *n* lines starting at the current line.	1	9999	:D3 Deletes three lines, starting with the current line.
:R*nm*	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 Repeats six 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. Use the key sequence Ctrl + G to access 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

Use this key sequence...	To move the insertion point....
Page Up	up a page
Home	to the beginning of the current line
Ctrl + Home or Ctrl + Page Up	to the beginning of the document
End	to the end of the current line
Ctrl + End or 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
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 pointer past the end of a line, the insertion point is placed immediately 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*

Text Selection	Instructions
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. 2 Still holding down the left mouse button, drag the mouse pointer within the margin to the last line that you want to select. 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. 2 Still holding down the left mouse button, drag the mouse pointer to the last character that you want to select. 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...	Instructions
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
Copy selected text	Press Ctrl + C or select **Edit** ⇨ **Copy**

To...	Instructions
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.
To highlight a column of code or data	Press the Alt key + the left mouse button. Drag to select.

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 772 .

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
Enter 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** option 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 Next**.

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
Enter 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
Enter 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** option 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 search
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 +].
- searches 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

Code Error Type	Instructions
Undefined keywords	Select **Tools** ⇨ **Options** ⇨ **Enhanced editor Appearance tab** in order to set the file elements **Defined keyword**, **User-defined keyword**, and the **Undefined keyword** to unique color combinations. When SAS recognizes a keyword, the keyword changes to the defined colors. You can 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. 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. Text that follows an open comment symbol (/*) remains the same color until the comment is closed with a closing comment symbol (*/).
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, select **Tools** ⇨ **Options** ⇨ **Enhanced Editor Options**. 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 136 .

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 **Indention** 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. The following table shows the keyboard shortcuts that you can use with bookmarking.

Note: Bookmarks are not saved with the file when the specific Enhanced Editor window is open.

Table 3.8 *Keyboard Shortcuts for Bookmarking Lines*

Action	Keyboard Shortcuts
Bookmark a line	Ctrl + F2 on an unmarked line
Unmark a line	Ctrl + F2 on a marked line
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 enter 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 my-v9-sas-files, which would expand to C:\Users \user-ID\Documents\My SAS Files\9.4';. 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, enter the name of the abbreviation.
3. In the **Text to insert for abbreviation** field, enter the text that the abbreviation expands into.
4. Click **OK**.

To use an abbreviation, enter 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. Enter 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**.

Using Word Tips

You can use Word Tips to help in understanding the abbreviations that you add.

To use Word Tips, you must first add abbreviations.

1 Press Ctrl + Shift + A.

2 Add the abbreviation and corresponding text.

3 Click **OK** to accept, or click **Cancel** to discard.

Word tips automatically appear as you type.

To accept a Word Tip, press Enter. To ignore the Word Tip, press Esc or continue typing.

To recall a Word Tip:

- Position the pointer so that it immediately follows the abbreviation.
- Press Alt + F1.

Submitting Your Program

You can submit either a complete program or a specified number of lines of your program, beginning with the first line.

Note: The maximum line length is 6K bytes.

Table 3.9 *Submitting a Complete Program*

Submitting a Complete Program	Instructions
When you open the program in the editor	Select the **Submit** check box.
From the Enhanced Editor	Do one of the following: - Click the **Submit** toolbar button - Press F3 or F8 - Select **Run** ⇨ **Submit** - 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.

Table 3.10 *Submitting a Portion of Your Program*

Submitting Partial Programs	Instructions
Only the top line of a program	Do one of the following: ■ Type **subtop** in the command bar. ■ Select **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. ■ Select **Run** ⇨ **Submit N Lines** and enter the number of lines that you want to submit.

For more information, see "SUBTOP Command: Windows" on page 414 .

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 133 .

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 the display.

```
data oranges;
   input variety $ flavor texture looks;
   total=flavor+texture+looks;
```

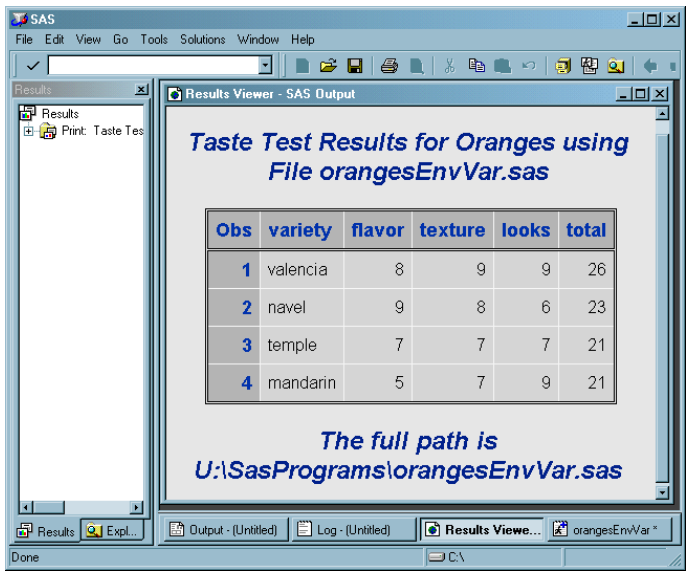
```
datalines;
navel 9 8 6
temple 7 7 7
valencias 8 9 9
mandarins 5 7 9
;

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:

Figure 3.2 *Using an Environment Variable to Place a Filename in DATA Step Output*



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, when you submit code in a batch session, or when you use the Program Editor.

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 can 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 772 provides a complete list of commands and their default keyboard shortcuts. The following table shows some of the more useful keyboard shortcuts.

Table 3.11 *Useful Keyboard Shortcuts*

Keyboard Shortcut	Task
Get help for a SAS procedure	Press 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

Keyboard Shortcut	Task
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 138 .

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 138 .

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 138 .

An alternative way to create a macro is to add commands by using the Create Keyboard Macro dialog box. To create the macro:

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. Enter the name of the macro in the **Keyboard macro name** field.
4. Enter 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 are 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 enables 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 selecting **Tools** ⇨ **Keyboard Macros** ⇨ **Macros**.

For Keyboard Macros only, click **Assign Keys** and choose a new shortcut key for each macro. Do not assign shortcut keys for Abbreviations.

To export a keyboard macro, click **Export**, select a folder from the **Save in** field, enter a filename in the **Filename** box, and click **OK**.

Note: You need to reassign the keyboard shortcut keys after importing the keyboard macros to avoid overwriting keyboard shortcut keys. The Abbreviations do not have key assignments.

Using Collapsible Code Sections

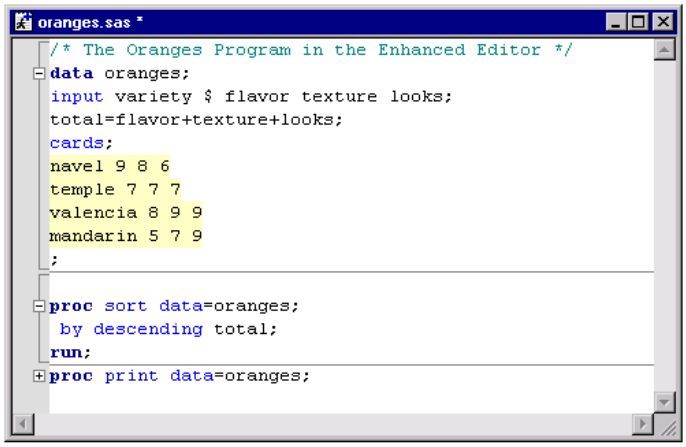
Collapsible code sections enable you to expand or collapse one or more sections of code. A section begins with a step keyword, a comment, or spaces above a section word or comment. A section ends with the next step keyword, a comment, or space above the next section word or comment. Step keywords include 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.

Note: The collapse and expand feature is not saved with the program, The feature is in effect only while the current editor window is open.

Figure 3.3 *The Enhanced Editor When Collapsible Code Segments Are Enabled*



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 positioning your pointer over the text or by using the keyboard selects only that line of text.
- Selecting text from above a collapsed section and pasting it 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 signature line of the collapsed section.
- 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.
- Entering something above a section that affects the section, such as comments or quotation marks, 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** does 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 3.12 *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

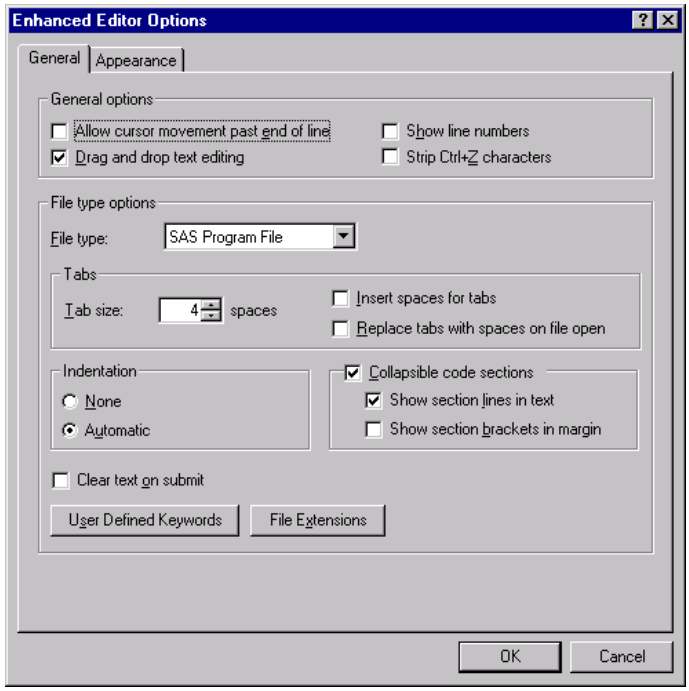
Opening the Enhanced Editor Options Window

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**.

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**.

Note: The enhanced editor options are stored in the Microsoft Windows Registry. The options are not stored in a file on the machine or in the SAS Profile catalog. The user must have operating system rights and permissions to edit the windows registry if they want to make changes.

Figure 3.4 *Editor Options Dialog Box*



General Editor Options

From the **General** tab that 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,by default, the insertion point is positioned immediately 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.

Indention
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 136 and "Using Schemes" on page 137 .

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 when 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.

Preview
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 You can also select a saved formatting scheme from the **Scheme** box. For more information about using schemes, see "Using Schemes" on page 137 .

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, font size, 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 enter 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

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 are assigned keyboard shortcuts, ensure that the **Show all commands** check box is not selected. To see all commands, including the commands 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 is displayed 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 772 for a list of the default keyboard shortcuts.

To reset keyboard shortcuts to the Enhanced Editor default:

1 Select **Tools** ⇨ **Options** ⇨ **Enhanced Editor Keys**.

2 Click **Assign keys** and then reset the key.

3 You can also click **Reset All** to reset all shortcuts.

Accessing the Enhanced Editor

By default, the Enhanced Editor is the active editor when you start SAS.

Access the Enhanced Editor by using the **Use Enhanced Editor** setting on the Preferences dialog box **Edit** tab or by issuing the WEDIT command. For more information, see "WEDIT Command: Windows" on page 427. If you do not want to access the Enhanced Editor when you start SAS, issue the NOENHANCEDEDITOR system option. For more information, see "ENHANCEDEDITOR System Option: Windows" on page 612.

If you do not access the Enhanced Editor, all Enhanced Editor windows remain available, and you can launch new Enhanced Editor windows by using the View menu or the WEDIT command. If the Enhanced Editor is not accessed when you start SAS, the Enhanced Editor window does not appear.

When the Enhanced Editor is accessed and you issue the Text Editor command in the **Tools** menu, the Enhanced Editor window appears. When the Enhanced Editor is not accessed, the Text Editor command opens SAS NOTEPAD.

Using the Program Editor

Overview of 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

Because the Enhanced Editor is the active editor by default when you start SAS, you must take one of the following actions to switch editors at start-up:

- Start SAS using the NOENHANCEDEDITOR system option.

- Instead of accessing the Enhanced Editor in the **Preferences** dialog box, use the menu to set the option:

  1 Select **Tools** ⇨ **Options** ⇨ **Preferences** ⇨ **Edit**.

  2 Deselect the **Use Enhanced Editor** check box.

  3 Click **OK**.

For more information about the NOENHANCEDEDITOR system option, see "ENHANCEDEDITOR System Option: Windows" on page 612 .

You can always access the Program Editor window 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 that 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 the default 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 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: Windows" on page 679 and "Changing the SAS Current Folder" on page 60 .

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 that 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 drag and drop a file:

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 that 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 quotation marks, in the **Filename** 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

Line numbers are turned off by default under Windows. You can enter **`numbers on`** in the command bar to display line numbers in the Program Editor window. You can also enter **`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 **View** tab.

Using Tabs

Many text editors retain tab characters. Other editors 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.

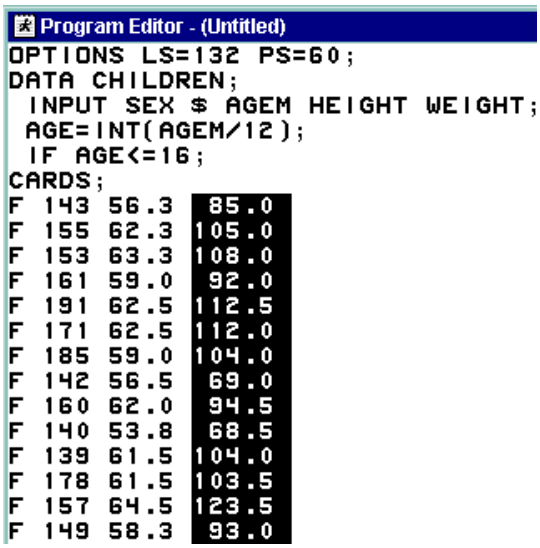
Understanding Line Breaks

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**. Here 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 that 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.)



- 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 entering text, the marked area is replaced with the new text. This action 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 79 .

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. Entering 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
Enter 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
Enter 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
Enter 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 3.13 *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 becomes 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 action is called nondefault drag and drop.

When you drag the selection to the target SAS window and release the mouse button, SAS displays a pop-up menu to let you choose which action to perform.

Table 3.14 on page 147 is a summary of drag-and-drop actions available for the possible target windows in SAS.

Table 3.14 *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 that 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 action 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 happens only 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 that 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 drive icon). If the file is to be saved for the first time, the Save As dialog box appears 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. Enter a filename in the **Filename** 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: Windows" on page 679 and "Changing the SAS Current Folder" on page 60 .

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 that 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 entering 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* turns on autosave using *minutes* as the interval.

For more information about the Autosave feature, see "Edit Preferences" on page 86 , "WAUTOSAVE Command: Windows" on page 421 , and "AUTOSAVELOC= System Option" in *SAS System Options: Reference*.

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 enables 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.

4

Using SAS Files under Windows

Introduction to SAS Files ... **152**
What Is a SAS File? ... 152
File Extensions for SAS Files ... 152
SAS Data Sets and Data Views (Member Types: Data and View) ... 154
SAS Catalogs (Member Type: Catalog) ... 156
SAS Stored Compiled DATA Step Programs (Member Type: Program) ... 156
Access Descriptor Files (Member Type: Access) ... 156

Multi Engine Architecture ... **157**
SAS Libraries ... 157
SAS Engines ... 157

Using Data Libraries ... **159**
Data Libraries ... 159
Specifying a Libref ... 159
Assigning SAS Libraries Using the Graphical User Interface ... 160
Assigning SAS Libraries Using the LIBNAME Statement or Function ... 160
Assigning SAS Libraries Using Environment Variables ... 163
Listing Libref Assignments ... 165
Clearing Librefs ... 166
Understanding How Multi-Folder SAS Libraries Are Accessed ... 167
Using the Sasuser Data Library ... 169
Using the Work Data Library ... 169
Using Large Data Sets with Windows and NTFS ... 171

Accessing SAS Files from Multiple SAS Sessions ... **171**

Using SAS Files from Other Versions with SAS 9.4 for Windows ... **172**
Introduction to Using SAS Files from Other Versions with SAS 9.4 for Windows ... 172
Using Release 6.08 through Release 8.2 Data Sets ... 174
Using Release 6.03 and Release 6.04 SAS Data Sets ... 174
Converting Release 6.08 through Release 6.12 SAS Data Sets ... 175
Using Version 7 and 8 Catalogs in SAS 9.4 ... 175
Converting Version 6 SAS Catalogs in SAS 9.4 ... 175
Converting Release 6.08 SAS Catalogs to SAS 9.4 ... 176
Converting Release 6.03 and Release 6.04 SAS Catalogs to SAS 9.4 ... 177
Creating Release 6.08 through Release 6.12 Data Sets ... 177

Using SAS 9.4 Files with Previous Releases ... **177**

Using Remote Host SAS Files in SAS 9.4 ... **178**

Reading BMDP, OSIRIS, and SPSS Files ... **178**

Overview of Reading BMDP, OSIRIS, and SPSS Files 178
BMDP Engine 179
OSIRIS Engine 180
SPSS Engine 181

Transferring SAS Files between Operating Environments **183**

Accessing Database Files with SAS/ACCESS Software **183**

Using the SAS ODBC Driver to Access SAS Data from Other Applications **184**

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 must be processed by SAS. SAS files are different from external files. External files can be processed by SAS statements and commands, they are not managed by SAS.

SAS files usually reside in SAS libraries. Under Windows, a SAS library is a named collection of SAS files within one or more Windows folders that SAS can access. The first time a file is accessed in the library, a SAS/ACCESS engine is associated with the library. The engine name specifies the access method that SAS uses to process the files in the library. SAS libraries are described in detail in "SAS Libraries" in *SAS Programmer's Guide: Essentials*.

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 Multi Engine Architecture. Multi Engine Architecture, combined with conversion utilities, provides access to SAS 9.4 files and SAS files created with previous releases of SAS (back to Version 5), whether they were created under Windows or other operating systems. Multi Engine Architecture also provides access to files created by other vendors' products, including database files. For more information, see "SAS Engines" in *SAS Programmer's Guide: Essentials*.

The following sections highlight information that 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 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 "Multi Engine Architecture" on page 157 . For more information about processing files from previous releases, see "Using SAS Files from Other Versions with SAS 9.4 for Windows " on page 172 .

Table 4.1 *Windows File Extensions and Their Corresponding SAS Member Types*

V6 File Extension	V7 and Beyond File Extensions	Short Extensions	SAS Member Type	Description
.sas	.sas	none	none	SAS program
.ss2	.sas7bpgm	ss7	Program	stored program (DATA step)
	.cfg (Version 8 and beyond)	none	none	configuration file
.lst	.lst	none	none	output file
.log	.log	none	none	log file
none	.sas7baud	st7	Audit	audit file
.sd2	.sas7bdat	sd7	Data	data set
.sv2	.sas7bvew	sv7	View	data set view
.si2	.sas7bndx	si7	Index	data set index. Indexes are stored as separate files but are treated by SAS as integral parts of the SAS data set.
.sc2	.sas7bcat	sc7	Catalog	SAS catalog
.sa2	.sas7bacs	sa7	Access	access descriptor file
.sf2	.sas7bfdb	sf7	FDB	consolidation database file
.sm2	.sas7bmdb	sm7	MDDB	multi-dimensional database file
none	.sas7bdmd	s7m	DMDB	data mining database file
none	.sas7bitm	sr7	ITEMSTOR	item store file

V6 File Extension	V7 and Beyond File Extensions	Short Extensions	SAS Member Type	Description
.su2	.sas7butl	su7	Utility	utility file
.sp2	.sas7bput	sp7	PUtility	permanent utility
.stx	none	none	none	transport file
none	.sas7bbak	none	none	backup file
	.spds9 (Version 9 and Beyond)	sp9	multiple	SAS SPD Engine component files and SAS SPD Server files

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: Do not delete files from your Work and Sasuser 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 WORKTERM system options when you start SAS. For more information, see "WORKINIT System Option" in *SAS System Options: Reference* and "WORKTERM System Option" in *SAS System Options: Reference*.

SAS Data Sets and Data Views (Member Types: Data and View)

The SAS data set is probably the most frequently used type of SAS file. SAS data sets 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 sets have a file extension of .sas7bdat.

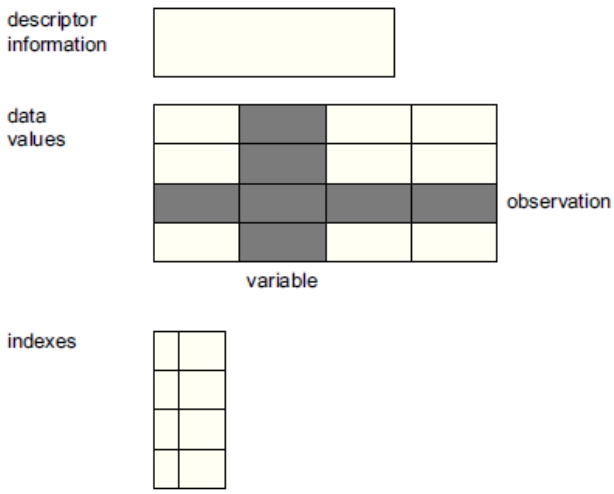
In SAS under Windows, SAS data sets can be indexed. The index is an auxiliary file that you create in order to provide fast access to records within a SAS data set through a variable or key. Indexes are stored as separate files, but are treated by

SAS as integral parts of the SAS data set. To learn more about indexes, see "Indexes" in *SAS V9 LIBNAME Engine: Reference*.

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. You can use the "CONTENTS Procedure Statement: Windows" on page 514 to access descriptor information.

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. For complete details,see "SAS Data Sets" in *SAS V9 LIBNAME Engine: Reference* and "SAS Data Sets" in *SAS Programmer's Guide: Essentials*.

Figure 4.1 *SAS Data Set Model*



You can think of extended attributes as customized metadata for your SAS files. Whereas common SAS attributes are predefined SAS system attributes, you can define extended attributes yourself. The DATA step preserves extended attributes from input data sets to output data sets. When it is saved on disk, this data set has a new extension .sas7bxat. You can create, add, delete, update, remove, and specify options for extended attributes using various XATTR statements in the DATASETS procedure. You can also use PROC CONTENTS to display data set and variable extended attributes. For more information, see "Extended Attributes" in *SAS V9 LIBNAME Engine: Reference*.

SAS 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.

A SAS 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 sets or SAS views. For more information, see "SAS Views" in *SAS V9 LIBNAME Engine: 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. An example of this action is the Sasuser.Profile catalog contains function 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.

For more information, see "Catalogs" in *SAS V9 LIBNAME Engine: Reference*.

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 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 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, see "Stored, Compiled DATA Step Programs" in *SAS V9 LIBNAME Engine: Reference*.

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 Interface to PC Files: Reference*, and other available SAS/ACCESS documentation.

Multi Engine Architecture

SAS Libraries

All permanent and temporary SAS files are stored in SAS libraries. A SAS library is a collection of SAS 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 library. Any Windows folder can be treated as a SAS library.

To use a SAS library in your SAS session, you must assign a libref (library reference) and an engine to the library. The libref is the name that 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 a variable or with the LIBNAME statement or function. For information about using librefs in Windows, see "Using Data Libraries" on page 159 . For a complete explanation of librefs, see "Elements of a Library Assignment" in *SAS Programmer's Guide: Essentials*.

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 Multi Engine Architecture.

The engine identifies the set of routines that SAS uses to access the files in the library. With this architecture, data can reside in different types of files, including SAS 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, see "SAS Engines" in *SAS Programmer's Guide: Essentials*.

When using the default engine, choose either V9 or BASE as the name that you will use in your SAS jobs now and in future releases. If your application is intended for SAS 9.4 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; 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 9.4 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 System Options window. Then select **Files** ⇨ **SAS Files**. The ENGINE system option displays the default engine for SAS libraries.

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 engine 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), SAS might be able to assign the libref to the appropriate 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, it 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 for SAS to 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: Windows” on page 610 .

Using Data Libraries

Data Libraries

The following sections provide information about data libraries, Librefs, Multi-Folder libraries, SAS user libraries, and Work 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 Start up** 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 Programmer's Guide: Essentials* 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 160 .
- Define a variable as described in "Assigning SAS Libraries Using Environment Variables" on page 163 .

Note: You can eliminate the LIBNAME statement by directly specifying the drive name and the data set name within quotation marks. 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 button , 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 appears, 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** dialog box 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 Functions and CALL Routines: Reference*.

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 DATA Step Statements: Reference*.

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 SAS 9.4 data sets are 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 9.4 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 bar 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 SAS files are located in multiple folders, you can treat these folders as a single SAS 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 library. When you concatenate SAS libraries, SAS uses a protocol for accessing the libraries, depending on whether you are accessing the libraries for read, write, or update.

You can 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 library whose libref is SALES.

For more information, see “Understanding How Multi-Folder SAS Libraries Are Accessed” on page 167 .

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 Options” on page 576 .

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 9.4 session, you can specify the V612 engine in the LIBNAME statement, as in the following example:

```
libname oldlib V612 'c:\sas612';
```

Another example is if you plan to share SAS files between SAS 9.4 under Windows and Version 6 under Windows, use the V6 engine when assigning a libref to the SAS library. Here is an example of specifying the V6 engine in a LIBNAME statement:

```
libname lib6 V6 'c:\sas6';
```

Remember that while SAS 9.4 can read Version 6 SAS data sets, Release 6 cannot read SAS 9.4 data sets. For methods of regressing a SAS 9.4 data set to a version 6 data set, see information in the Migration focus area at **http:\\support.sas.com\migration\planning\files\regression.html.**

For more information about using engine names in the LIBNAME statement, see “Using SAS Files from Other Versions with SAS 9.4 for Windows ” on page 172 and “Reading BMDP, OSIRIS, and SPSS Files” on page 178 . You can also see the LIBNAME statement in *SAS DATA Step Statements: Reference*.

Making Librefs Available When SAS Starts

Instead of assigning the same librefs each time you start SAS, you can specify a libref each time that SAS starts. In the **New Library** dialog box, select **Enable at start-up**. The libref is available as soon as SAS initializes. Libraries that are enabled at start-up 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 engines, only those SAS files that were created with the V6 engine that is assigned to the given libref can be accessed by using that libref. SAS files created with the V7, V8, and V9 engines can all access each other, but those engines cannot access files created with the V6 engine. You must specify the V6 engine to see those files. You can assign multiple librefs with different engines to a folder. For example, the following statements are valid:

```
libname one V6 '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 library. For example, the following two LIBNAME statements assign the librefs MYLIB and INLIB (both using the V9 engine) to the same SAS 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 variables instead of the LIBNAME statement or function. A variable equates one string to another. SAS recognizes two types of variables:

- SAS variables
- Windows 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 that takes precedence
2. a libref assigned by a SAS variable
3. a libref assigned by a Windows variable.

For example, if the Windows variable TEMP is assigned to C:\Windows\TEMP and you use the following LIBNAME statement:

```
libname temp c:\public
```

the LIBNAME resolves to `c:\public`.

There are two ways of defining a variable to SAS:

- Use the SET system option. This option defines a SAS (internal) variable.
- Issue a Windows SET command. This command defines a Windows (external) variable. Alternatively, under Windows, you can define 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 variables. If you use variables as librefs, you must accept the default engine.

The availability of variables makes it simple to assign resources to SAS before invocation.

Using a SAS Environment Variable as a Libref

You can use the SET system option to define a SAS 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 variable TEST to this SAS library:

```
-set test c:\sas\mysasdata
```

When you assign a variable, SAS does not resolve the reference until the variable name is actually used. For example, if the TEST variable is defined in your SAS configuration file, the 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 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: Windows” on page 682 .

Any variable name that you use as a value for a system option in your SAS configuration file must be defined as a variable before it is used. For example, the following SET option must appear before the SASUSER option that uses the variable TEST:

```
-set test "d:\mysasdir"
-sasuser "!test"
```

In the following example, 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 variables in your SAS configuration file or in a LIBNAME statement in your SAS programs, you must precede the 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 before invoking SAS to create a Windows variable. You must define the variable before invoking SAS; you cannot define variables for SAS use from a Command Prompt window from within a SAS session.

SAS can recognize variables only if they have been assigned in the same context that invokes the SAS session. You must define the 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 a variable in a Command Prompt window, and then start SAS from the Start menu (or with another shortcut), SAS does not recognize the variable.

The 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 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 variables that you have defined as librefs are listed, provided you have used them in your SAS session. If you have defined a variable as a libref but have not used it yet in a SAS program, the 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 Explorer 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

Overview of Clearing Librefs

You can clear a libref by using one of the following methods:

- "SAS Explorer Window" on page 166
- "LIBNAME Statement" on page 167
- "LIBNAME Function: Windows" on page 483

SAS automatically clears the association between librefs and their respective libraries at the end of your job or session. If you want to associate the libref with a different SAS 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 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 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 appears.
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 that 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 a variable, the variable remains defined, but it is no longer considered a libref, and 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 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 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: C:\Users*user ID*\Documents\My SAS Files\9.4

You can use the SASUSER system option to make the Sasuser libref point to a different SAS 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 9.4 files.

For more information about your Profile catalog, see “Profile Catalog ” on page 36 . For more information about the SASUSER system option, see “SASUSER System Option: Windows” on page 680 .

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_nodename_*, as discussed in “Work Data Library ” on page 38 . 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 might 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 Explorer window now, you see the Sports data set in the Work folder.

You can display all the temporary data sets that are created during this session from these locations:

- the 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 9.4 files.

Using an Environment Variable

You can use a 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 a variable to use for your Work data library:

```
-set myvar c:\ tempdir
-work !myvar
```

The SET option associates the MYVAR 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 that 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 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 variable named USER. If you have a Windows 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: Windows" on page 711 and *SAS System Options: Reference*.

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. Two terabytes is the practical limit for physical and logical volumes using NTFS.

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 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: Windows" on page 672 enables 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 9.4 for Windows

Introduction to Using SAS Files from Other Versions with SAS 9.4 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 9.4 format.

SAS 9.4 file formats are the same as Version 7 and 8 file formats with the exception of a few new features. Table 4.2 on page 173 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.

If you want to use SAS 9.4 to access catalogs that were created with earlier releases of SAS for Windows, you might have to convert the catalogs from the earlier releases to the SAS 9.4 format before you can use the catalogs in a SAS 9.4 program.

The following table provides information about data set and catalog conversion.

Table 4.2 *Summary of Using Version 6, 7, and 8 Data Sets and Catalogs in SAS 9.4*

Version or Release	Data Sets	Catalogs
Version 7 and 8	In 32-bit SAS, no action is necessary. SAS reads, updates, and writes to Version 7 and Version 8 data sets. In 64-bit SAS there are no updates to read and write. No action is required.	In 32-bit SAS no action is necessary. SAS reads, updates, and writes to Version 7 and Version 8 catalogs. In 64-bit SAS, migrate to SAS 9 by using the MIGRATE procedure with the SLIBREF option.
Releases 6.08 - 6.12	The V6 engine is automatically detected. In 32-bit SAS no action is necessary. SAS reads, updates, and writes to Version 6 data sets. In 64-bit SAS, SAS can read a V6 data set but cannot write to a V6 data set.	In 32-bit SAS, SAS can read a Version 6 catalog but cannot write to it. In 64-bit SAS convert to SAS 9 by using the CPORT and CIMPORT procedures.
Releases 6.03 and 6.04	Use the V604 engine to read data. You cannot write to Release 6.03 and 6.04 data sets.	not supported

As the table shows, in 32-bit SAS, except for Release 6.04 and Release 6.03 data sets, Version 6 (32-bit SAS) and Version 7 and 8 data sets do not need to be converted to SAS 9.4 data sets in order for SAS 9.4 to read, update, and write to the data sets. In 64-bit SAS, the cross- data access (CEDA) facility imposes some restrictions.

In 32-bit SAS, Version 7 and 8 catalogs also do not need to be converted to V9 catalogs. Version 6 SAS catalogs can be read but not updated. If a Version 6 catalog is to be updated, you must convert it to a SAS 9.4 catalog.

The Migration focus area at **`http://support.sas.com/migration`** 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 under an operating system other than Windows, you need to transport those files to Windows. A separate document, *Moving and Accessing SAS Files*, discusses transporting files from one operating system to another operating system.

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 in the LIBNAME statement. The default engine is 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:

```
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, suppose you are unsure of the file's version or suppose you know that the SAS library is a mixed mode library. In those cases, you must specify the engine name in the LIBNAME statement in order 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 174 .

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 32-bit Windows SAS 9.4 session. Release 6.03 and Release 6.04 SAS data sets are not compatible with the x64 64-bit SAS. (Remember that there is no difference between Release 6.03 and Release 6.04 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 9.4 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 9.4 session:

```
libname mylib v604 'c:\sas604';
proc print data=mylib.fruit;
run;
```

Converting Release 6.08 through Release 6.12 SAS Data Sets

You should convert your Version 6 SAS data sets to the SAS 9.4 format if you access them often and do not need to read the files from Version 6. The data set format of SAS 9.4 is more efficient than the Version 6 format, and there are new SAS 9.4 features that cannot be used unless the data sets are converted. You should migrate Release 6.12 libraries by using the MIGRATE procedure. For information about how to convert libraries before 6.12, see *Moving and Accessing SAS Files*.

Note: For more information about conversion, you can access information from the Migration focus area: `http://support.sas.com/migration`.

Using Version 7 and 8 Catalogs in SAS 9.4

Because SAS 9.4 file formats are basically the same as Version 7 and 8 file formats, a 32-bit platform can read, update, and write to Version 7 and 8 catalogs without having to convert them to SAS 9.4 catalogs.

However, when SAS is running under 64-bit Windows, it cannot read 32-bit catalogs.

Converting Version 6 SAS Catalogs in SAS 9.4

Because of the differences in the internal structures of the operating s, you can use the CPORT and CIMPORT procedures to convert Version 6 SAS catalogs that were created under Windows to SAS 9.4 format before you can use the catalogs in your SAS 9.4 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 disk) to a location that your SAS 9.4 session can read.
3. Use the CIMPORT procedure from your SAS 9.4 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 the *Base SAS Procedures Guide*.

There are other conversion methods. For information see *Moving and Accessing SAS Files*.

Converting Release 6.08 SAS Catalogs to SAS 9.4

If you are converting directly from Release 6.08 to SAS 9.4, you can use the CPORT procedure in Release 6.08 to create a transport file, and then use the SAS 9.4 CIMPORT procedure to convert the catalog to a SAS 9.4 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 9.4. (The C16PORT procedure is not available in SAS 9.4.)

To convert your SAS catalogs from Release 6.08 under Windows to SAS 9.4:

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 9.4 can read it.
3. Use the 9.4 CIMPORT procedure to read the transport file and create a converted SAS V9.4 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 9.4 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, and 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 transport 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 9.4. For information about the CPORT and CIMPORT procedures, see the *Base SAS Procedures Guide* and *Moving and Accessing SAS Files*.

Converting Release 6.03 and Release 6.04 SAS Catalogs to SAS 9.4

If you want to convert Release 6.04 SAS catalogs to their SAS 9.4 counterparts, see *Moving and Accessing SAS Files*.

Creating Release 6.08 through Release 6.12 Data Sets

You might need to create Release 6.08 through Release 6.12 data sets from your SAS session under Windows. This action 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 that is 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 9.4 Files with Previous Releases

Do not use the CPORT and CIMPORT procedures for regressing a SAS file to a previous release. You experience errors when you try to import the transport file. For more information about transporting files, see *Moving and Accessing SAS Files*.

If a SAS 9 file is created under a 32-bit platform for Windows, then the file is fully compatible in a Version 7 or 8 session as long as you do not use any new SAS 9 file features that are not supported under Version 7 or 8. SAS 9 files are not supported under Version 6 but are usable after regression to Version 6 format.

To learn about these compatibility issues and the best methods for regressing a SAS 9 file to a previous release, see “Using a SAS 9 File Under a Previous Version of

SAS" in the Migration focus area at **http://support.sas.com/migration/planning/files/regression.html**.

Using Remote Host SAS Files in SAS 9.4

SAS 9.4 can read Version 7 or later SAS data sets created on a directory-based host such as Windows or Linux. The file must exist in a hierarchical file structure. The file can be accessed from a Network File Services (NFS) mounted drive or copied to Windows with File Transfer Protocol (FTP) in binary mode. CEDA processing is invoked. Alternatively, you can create a transport data set and transport your files from the host to Windows.

Note: The Solaris platform is no longer available starting in SAS 9.4M9. Documentation remains for existing customers who have not upgraded to SAS 9.4M9.

A complete explanation of using remote host SAS files in SAS 9.4 can be found in *Moving and Accessing SAS Files*. For cross-release compatibility, see the Migration focus area at **http://support.sas.com/migration**.

Reading BMDP, OSIRIS, and SPSS Files

Overview of Reading BMDP, OSIRIS, and SPSS Files

SAS 9.4 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. When using BMDP and OSIRIS engines, 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. When using the SPSS engine, PROC COPY or a DATA step supports the portable file format. 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 action 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 sets. For more information, see "CONVERT Procedure Statement: Windows" on page 516 .

- "BMDP Engine" on page 179
- "OSIRIS Engine" on page 180
- "SPSS Engine" on page 181

BMDP Engine

Overview of the BMDP Engine

The BMDP interface library engine enables you to read BMDP DOS files from the BMDP statistical software application 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 convention 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, and 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

Overview of the 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 Statement: Windows" on page 516 .

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 action assumes that you downloaded the OSIRIS files from your host computer in binary format. The following section assumes that you are familiar with the OSIRIS file terminology. [1]

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:

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 a variable name or a fileref. (Do not use quotation marks if it is a 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 systems. The reason is that the OSIRIS software does not run outside the z/OS , but the engine is designed to accept an z/OS data dictionary on any other operating 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 controlled by the contents of the OSIRIS dictionary file.

1. See documentation provided by the Institute for Social Research for more information.

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

Overview of the 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. [1]The SPSS engine is a read-only engine.

To read an SPSS export file that 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 .

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_ .)

Note: SPSS can have system-missing and user-defined missing data. When you use the SPSS engine or PROC CONVERT, the missing values (user-defined or system) are converted to system-missing values. User-defined missing values have to be recoded as valid values. When the data set is converted, you can use PROC FORMAT to make the translation (for example, –1 to .A and –2 to .B).

1. 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 portable file:

```
libname mylib spss 'myspss.por';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

Reformatting SPSS Files

SAS cannot use an SPSS file that contains a variable that has a numeric format that has a larger number of decimal places than the width of the entire variable. For example, if an SPSS file has a variable that has a width of 17 and also has 35 decimal places, SAS returns errors when you try to run a DATA step on the file or when you try to view it with the table viewer. To use the SPSS file with SAS, you must reformat the variable.

You can reformat the variable by reducing the number of decimal spaces to a value that fits within the width of the variable. In the following code, the statement `revision=cat(format,formatl,'.2');` converts the number of decimal spaces to 2. This value reduces the number of decimal spaces so that it is not greater than the width of the variable.

```
libname abc spss 'FILENAME.POR';
proc contents data=abc._all_ out=new; run;
filename sascode temp;
data _null_; set new; file sascode;
     if formatd > formatl then do;
        revision=cat(format,formatl,'.2');
        put 'format' +1 name +1 revision ';' ;
        end;
     run;
data temp; set abc._all_;
     %inc sascode/source2;
     run;
```

Note: The OPTIONS NOFMTERR statement does not allow SAS to use a data set that has a DATA step or the table viewer. You must reformat numeric variables that have a larger decimal space value than their width before you can use a DATA step or the table viewer.

Transferring SAS Files between Operating Environments

For more information about transferring SAS files between operating environments, see "Deciding to Move a SAS File between Operating Environments" in *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 as if it were a SAS data set.

SQL Procedure pass-through facility
: accesses data from several 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 Interface to 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 about configuring and using the SAS ODBC Driver, see *SAS Drivers for ODBC: User's Guide*.

5

Using External Files under Windows

About External Files ... 185

Referencing External Files ... 186
- Accessing External Files ... 186
- Using a Fileref ... 187
- Using a Quoted Windows Filename ... 197
- Using a File in Your Working Directory ... 198
- Running External LUA Files ... 198

Accessing External Files with SAS Statements ... 198
- Overview of Accessing External Files with SAS Statements ... 198
- Using the FILE Statement ... 199
- Using the INFILE Statement ... 200
- Using the %INCLUDE Statement ... 201

Accessing External Files with SAS Commands ... 201
- Overview of Accessing External Files with SAS Commands ... 201
- Using the FILE Command ... 202
- Using the INCLUDE Command ... 202
- Using the GSUBMIT Command ... 203

Advanced External I/O Techniques ... 204
- Overview of Advanced External I/O Techniques ... 204
- Altering the Record Format ... 204
- Appending Data to an External File ... 205
- Determining Your Drive Mapping ... 205
- Reading External Files with National Characters ... 206

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 Programmer's Guide: Essentials contains basic, platform-independent information about external files.

For **information** about 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 535 , and in the SAS statements section in *SAS DATA Step Statements: Reference*.

You can also specify external files in various SAS dialog box entry fields (for example, as a file destination in the Save As dialog box), the FILENAME function, and in SAS commands, such as FILE and INCLUDE.

Depending on the context, SAS can reference an external file by using the following information:

- 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 quotation marks
- 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 quotation marks.
2. Check for a fileref defined by a FILENAME statement or function.
3. Check for an environment variable fileref.
4. Assume that the file is in the working directory.

In other words, SAS assumes that an external file reference is a standard Windows file specification. If it is not, SAS checks to determine whether 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 filerefs, 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

Overview of 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. You can also check **Enable at Start up** 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-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 Functions and CALL Routines: Reference*.

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, FILE, and INFILE statements 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 192 and "Assigning a Fileref to Concatenated Files" on page 193 .

The * and ? wildcards can be used in either the external filename or file extension for matching input filenames. 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 204 . For the complete syntax of the FILENAME statement, refer to "FILENAME Statement: Windows" on page 542 .

Using Environment Variables

Just as you can define an environment variable to serve as a logical name for a SAS library (see "Assigning SAS Libraries Using Environment Variables" on page 163), 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 before invocation. However, the environment variables that 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 does 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 that you are referencing, as long as the file extension is the expected

one. For example, 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 filename 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 quotation marks, 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 that 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 191 .

Another feature of member-name syntax is that it enables you to reference a subdirectory in the working directory without using a fileref. For 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 directory and the D:\MYPROGS directory. When you use the fileref PROGS in your SAS program, SAS looks in these directories for the member that 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 196 .

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 191 . 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 might 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 196 .

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 192 .

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 box, 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 quotation marks to reduce ambiguity (since a single quotation mark is a valid character in a long filename). When you need to specify multiple filenames, enclose each filename in double quotation marks 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 **File Shortcuts**. The Explorer window lists all the filerefs active for your current SAS session. Any environment variables that 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 that you have assigned during your current SAS session are cleared.

To clear filerefs using the SAS Explorer File Shortcuts:

1. select the File Shortcuts that 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 on 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 enter 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 entered. For example, if you issue the following statements, the %INCLUDE statement submits four 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

Overview of 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 that 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 that you want to reference is located in the working directory. For example, if in the previous example the working directory is C:\MYDIR, you can submit this statement:

```
%include "oranges.sas";
```

Using a File in Your Working Directory

If you store the external files that you need to access in your working directory and they have the expected file extensions (see Table 5.1 on page 191), you can simply refer to the filename, without quotation marks or file extensions, in a SAS statement. For example, if a filename ORANGES.SAS is 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 24 and “Changing the SAS Current Folder” on page 60 .

Running External LUA Files

You can run external scripts written in the LUA programming language from the SAS command line. For more information, see “Running External Lua Files” in *SAS Companion for UNIX Environments*.

Accessing External Files with SAS Statements

Overview of 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 204 .

- “Using the FILE Statement” on page 199

- "Using the INFILE Statement" on page 200
- "Using the %INCLUDE Statement" on page 201

Using the FILE Statement

The FILE statement enables you to direct lines that are written by a PUT statement to an external file.[1]

Here is an 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 equal sign (=), and the variable value are all written to the file. For more information about named output, see the description of the PUT statement in *SAS DATA Step Statements: Reference*.

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 204 . For the complete syntax of the FILE statement, see "FILE Statement: Windows" on page 538 .

1. 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 DATA Step Statements: Reference*.

The default record length that is used by the FILE statement is 32767 characters. If the data that you are saving contains records that are longer than 32767 characters, 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 "FILE Statement: Windows" on page 538 .

You can also specify a different value, instead of the default 32767, for LRECL= in an OPTIONS statement or in your configuration file. This value stays in effect during the entire session. If you want to specify a different LRECL= value for a specific step, then you must specify the value in a FILENAME, FILE, or INFILE statement.

Using the INFILE Statement

Use the INFILE statement to specify the source of data that is 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:

```
filename mydata "c:\mysasdir\survey.dat";
data survey;
   infile mydata;
   input fruit $ taste looks;
run;
```

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 204 . For the complete syntax of the INFILE statement, see "INFILE Statement: Windows" on page 552 .

The default record length that is used by the INFILE statement is 32767 characters. If the data that you are reading has records longer 32767 characters, 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 "INFILE Statement: Windows" on page 552 .

You can also specify a different value, instead of the default 32767, for LRECL= in an OPTIONS statement or in your configuration file. This value stays in effect during the entire session. If you want to specify a different LRECL= value for a specific step, then you would need to specify the value in a FILENAME, FILE, or INFILE statement.

Using the %INCLUDE Statement

When you submit an %INCLUDE statement, it reads an entire file into the current SAS program that 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 that are 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 204 . For the complete syntax of the %INCLUDE statement, see “%INCLUDE Statement: Windows” on page 549 .

The default record length used by the %INCLUDE statement is 32767 characters. If the program that you are reading has records longer than 32767 characters, 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 “%INCLUDE Statement: Windows” on page 549 .

You can also specify a different value, instead of the default 32767, for LRECL= in an OPTIONS statement or in your configuration file. This value stays in effect during the entire session. If you want to specify a different LRECL= value for a specific step, then you would need to specify the value in a FILENAME, FILE, or INFILE statement.

Accessing External Files with SAS Commands

Overview of 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 filename 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 191.

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 action 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 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 “Saving Files” on page 111 and “Using the Program Editor” on page 140.

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 204. For the complete syntax of the FILE command, see “FILE Command: Windows” on page 404.

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 "Opening Files" in "Saving Files" on page 111 and "Using the Program Editor" on page 140.

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 204. For the complete syntax of the INCLUDE command, see "INCLUDE Command: Windows" on page 410.

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 are not 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

Overview of 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" on page 204
- "Appending Data to an External File" on page 205
- "Determining Your Drive Mapping" on page 205
- "Reading External Files with National Characters" on page 206

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 contains 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 that you specify.

Appending Data to an External File

Occasionally, you might 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 Functions and CALL Routines: Reference* 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 the information in Output 5.1 on page 206 .

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=32767
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. Currently, 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 Word) 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.

6

Managing SAS Output under Windows

Printing ... **208**
Introduction to Printing in SAS within the Windows Environment ... 208
Printing from within a SAS Window ... 208
Previewing Your Output Before You Print ... 218
Using SAS Print Forms ... 220
Printing with SAS Commands ... 222
Sending DATA Step Output to a Printer ... 222
Sending Printed Output to a File ... 223
Printing in Batch Mode ... 224
Default Printer Details ... 224
Canceling a Print Job ... 224

Routing Procedure Output to a Web Browser ... **225**
Introduction to Routing Procedure Output to a Web Browser ... 225
Configuring Preferences for HTML Output ... 226
Using the Results Viewer Window ... 227

Routing Procedure Output and the SAS Log to a File ... **228**
Introduction to Routing Procedure Output and the SAS Log to a File ... 228
Using the Save As Dialog Box ... 229
Using the PRINTTO Procedure ... 229
Using SAS System Options ... 230

Using the SAS Logging Facility to Write Log Messages ... **231**

Producing Graphics ... **232**
Producing Graphics on Your Display ... 232
Printing Graphics ... 233
Importing Graphics from Other Applications ... 237
Exporting Graphics for Use with Other Applications ... 239
Additional Resources ... 244

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 Programmer's Guide: Essentials* and "UPRINTMENUSWITCH System Option: Windows" on page 710 .

For details about using Windows printing, see your Windows documentation. "Producing Graphics" on page 232 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

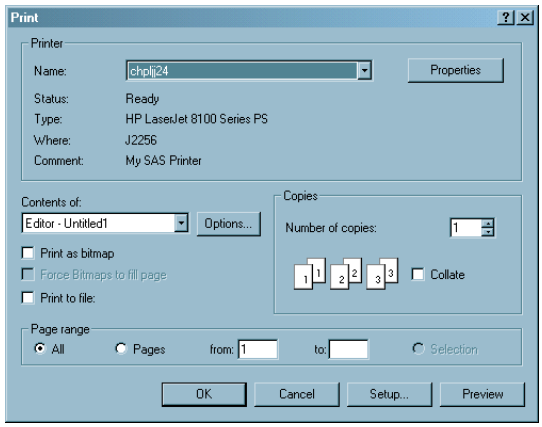
Use the **Print** dialog box to set the following print options:

- 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**

Figure 6.1 *Print Dialog Box*

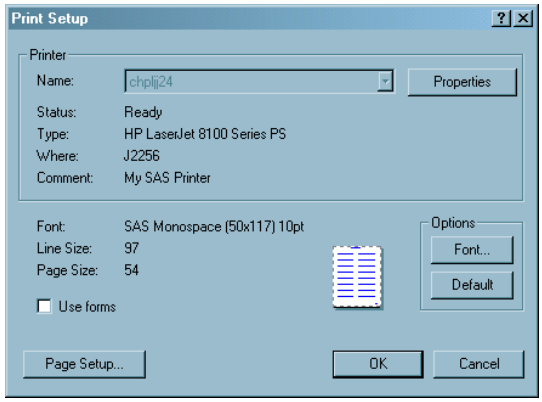


Use the **Print Setup** dialog box to set the following print options:

- Change fonts
- Use Forms.

To access the **Print Setup** dialog box, select **File** ⇨ **Page Setup**

Figure 6.2 *Print Setup Dialog Box*



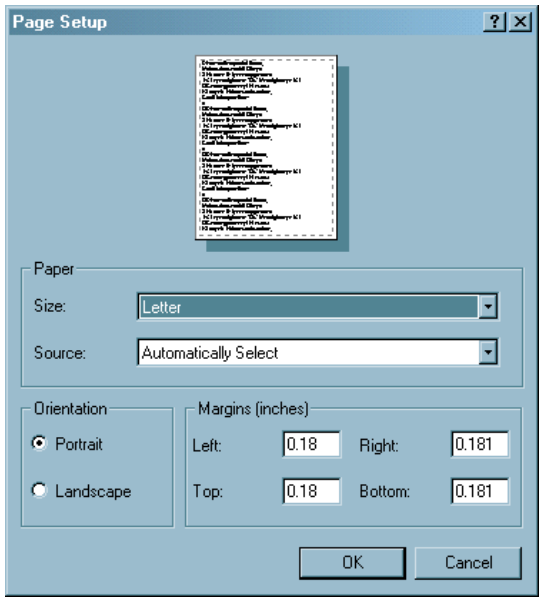
Use the **Page Setup** dialog box to set the following print options:

- 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**

Figure 6.3 Page Setup Dialog Box



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 whether 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: Windows" on page 703 and "PRTPERSISTDEFAULT System Option: Windows" on page 667 .

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 Enter 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 Options 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 Enter 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: Windows" on page 705 .

Setting Up the Printed Page

Setting up a page involves specifying the paper, the orientation of the paper, and the margins. You can 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.	Enter 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. 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 left mouse button.
3. Enter 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

Note: 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 specified in the Windows print dialog boxes.

Support for a particular paper size is printer dependent. Here 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 ■ paper source ■ orientation ■ margin settings	■ font ■ font style ■ size

Although you cannot set the line size and page size 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 dialog boxes 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	Select **Print as bitmap** In the **Contents of** box, select **AWS windows (bitmap)**.
Print the entire screen	Select **Print as bitmap** 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 enter 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.

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, enter 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**.
  - Enter the beginning page in the **from** box.
  - Enter 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 appears as printed output:

1. Select the window that 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 776 .

Print Preview Shortcut Keys

In the Print Preview window, you can navigate by using the following shortcut keys:

Table 6.6 *Print Preview 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

Key	Action in Full Page Mode	Action in Zoom Mode
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: Windows" on page 668 .

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 line size, page size, 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 bar.

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.

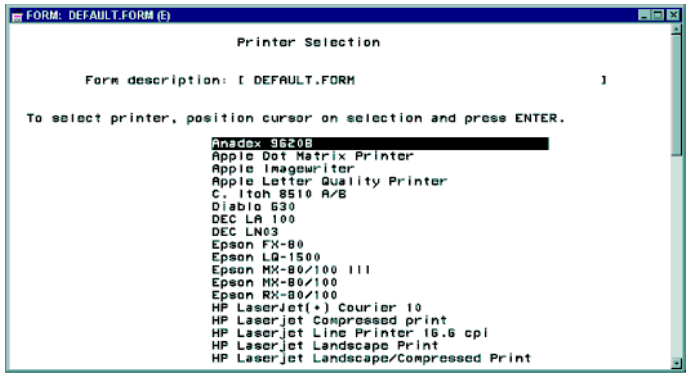
Although the majority of the frames in the FORM window are the same across all operating environments, the first frame that you see after issuing the FSFORM command is the Printer Selection frame, which lists the printers that you are able to use under Windows. Figure 6.4 on page 221 shows the default information for this frame.

To navigate through the FORM window frames for a printer:

1 Click a printer name or select a printer and press Enter.

2 Select **Tools** ⇨ **Next Screen** ⇨ **Previous Screen** to move through the frames.

Figure 6.4 *Printer Selection Frame*



The information in the Printer Selection frame is also site-dependent, so the printer list at your site is different from the one shown in Figure 6.4 on page 221 .

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 entering 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 about these commands, refer to the SAS Help and Documentation.

These SAS commands open the print dialog boxes:

Table 6.7 *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 can 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 action, 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: Windows” on page 703 .

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 action 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 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: Windows" on page 703 . 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 211 .

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.

Table 6.8 *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: Windows" on page 666 .

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 Microsoft Internet Explorer is 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 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 enter a path in the **Folder** text box or click **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 "Overview" in *SAS Output Delivery System: Procedures Guide* 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 Microsoft Internet Explorer is not installed, you must select **Preferred web browser**.

To configure the **Web** page of the **Preferences** dialog box:

- Click the **Web** tab.
- Choose which browser you want to use:
  - Select the **Use default browser** 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 browser** radio button to view HTML from a web browser other than the default browser. Then enter 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. If you want to see your procedure output as it is generated, 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 186 .

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 which 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 111 within the Enhanced Editor or "Saving Files" on page 148 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 can 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 action affects how 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 Statement: Windows" on page 527 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: Windows" on page 589 and "ALTLOG System Option: Windows" on page 588 .

Note: The Status window does reflect the PRINT and LOG system options values when recording where the procedure output and log are being sent.

Using the SAS Logging Facility to Write Log Messages

The SAS logging facility makes it possible to categorize, collect, and filter log events and write them to a variety of output devices. The logging facility supports problem diagnosis and resolution, performance and capacity management, and auditing and regulatory compliance. The logging facility has following features:

- Log events are categorized using a hierarchical naming system that enables you to configure logging at a broad level or a fine-grained level.
- Log events can be directed to multiple output destinations, including files, operating system facilities, databases, and client applications. For each output destination, you can specify the following information:
  - the categories and levels of events to report
  - the message layout, including the types of data to be included, the order of the data, and the format of the data
  - filters based on criteria such as diagnostic levels and message content
- Logging diagnostic levels can be adjusted dynamically without starting and stopping processes.
- Performance-related events can be generated for processing by the Application Response Measurement (ARM) 4.0 server.

The logging facility is used by most SAS server processes. You can also use the logging facility within SAS programs.

For Windows, log messages can be written to the Windows event viewer.

For information about using the logging facility in your operating environment, see *SAS Logging: Configuration and Programming Reference*.

Producing Graphics

Producing Graphics on Your Display

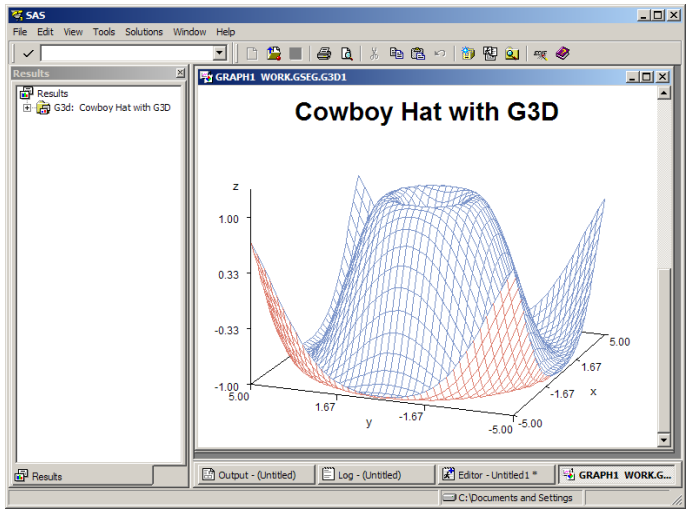
In most cases, output is automatically displayed on your monitor when you run a SAS/GRAPH procedure; it is not necessary to specify a SAS/GRAPH device driver. Information about your graphics display is stored in a Windows information file and is automatically used by SAS during an interactive SAS session.

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;
   title 'Cowboy Hat with G3D';
run;
quit;
```

The following display shows the output for this program:

Figure 6.5 *Cowboy Hat Program Output*



If you use the DEVICE= option in the GOPTIONS statement to route your graphics to a hard-copy 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

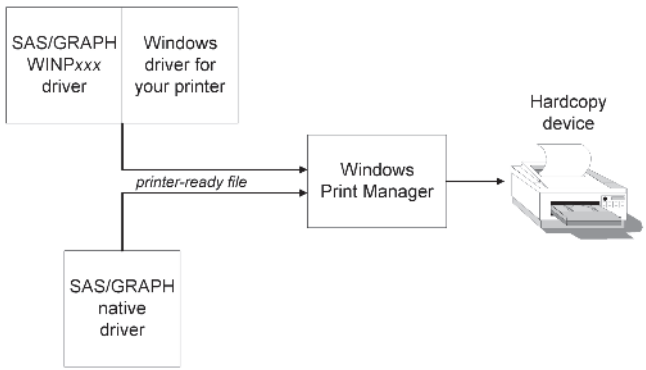
Overview of Printing Graphics

You can use two methods to print output from SAS/GRAPH:

- "Using the SAS/GRAPH Generic (WINPxxx) Drivers" on page 234 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.
- "Using the SAS/GRAPH Native Printer Drivers" on page 235 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 drivers 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 6.6 on page 234 illustrates how you can use the two sets of printer drivers within SAS/GRAPH to produce output for a given device.

Figure 6.6 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 WINPxxx Driver" on page 236 . 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.

For more information, select the following topics:

- "Using the SAS/GRAPH Generic (WINPxxx) Drivers" on page 234
- "Using the SAS/GRAPH Native Printer Drivers" on page 235
- "Printing and Previewing from the GRAPH Window" on page 236
- "Choosing between a SAS/GRAPH Native Driver and the WINPxxx Driver" on page 236

Using the SAS/GRAPH Generic (WINPxxx) Drivers

To print a graphic by using the SAS/GRAPH generic device drivers with the Windows printer drivers:

1 Select **File** ⇨ **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 or 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 can 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 that you specify*xxx* depends on the type of output device that you use to print your graph:

WINPRTM
for black and white (monochrome) printers

WINPRTG
for gray scale 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, you must change the top and bottom margins by the same amount that 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 in 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** menu and then select **Print**.

Note: If you do not specify a target device before you create the graph, SAS/GRAPH prompts 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 that you specify.

Choosing between a SAS/GRAPH Native Driver and the WINPxxx 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 hard-copy 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 WINPxxx 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 WINPxxx 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 765 .

Importing Graphics from Other Applications

Overview of Importing Graphics from Other Applications

SAS/GRAPH lets you import bitmap and vector graphics that were created by other software applications. SAS/GRAPH provides these benefits:

- You can create your graphic using another graphics editor. 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 238 .
- 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 238 .

- "Pasting Graphics from the Windows Clipboard" on page 238
- "Importing a Graphics File from within a SAS/GRAPH Window" on page 238

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 6.9 *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

Graphics File Format	File Extension
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. The **Format** field must show the correct source format; the field indicates which import filter SAS/GRAPH uses. 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

Overview of 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 240 .
- 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 241 .
- 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 242 .
- 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 244 .

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* 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 6.10 *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

Graphics File Format	File Extension
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 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 position the pointer 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 copies 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 action 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, the graph retains its separate components so that you can independently edit and size it. The editing capabilities that 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. CGMOFML and CGMOFMP are the recommended device drivers for all CGM output. If CGMOFML and CGMOFMP are not adequate for the receiving software, then you can use the devices in Table 6.11 on page 242 . This table lists the graphics packages to which you can export CGMs and the appropriate drivers to use.

The driver names that are marked with an asterisk (*) are already provided with SAS 9.4. Some of the drivers have been disabled and are designated, in the table, by the phrase, disabled in device catalog. To enable a device driver, do the following:

1 Point the library GDEVICE0 to a new location. For example:

```
LIBNAME GDEVICE0 'directory';
```

2 Use PROC CATALOG to copy some or all of the entries to the GDEVICE0 location. For example:

```
PROC CATALOG C=SASHELP.DGDEVICE;
COPY OUT=GDEVICE0.DEVICES;
RUN;
```

For more information about how to use the CGM drivers and graphics packages, contact SAS Institute's Technical Support Division.

Table 6.11 *CGM Drivers for Popular Graphics Packages*

Package	Suggested Driver
Aldus PageMaker	CGMAPMA* (disabled in dgdevice catalog)
Aldus Persuasion	CGMAPSA* (disabled in dgdevice catalog)
BPS 35 MM Express	CGM35 (disabled in dgdevice catalog)

Package	Suggested Driver
Borland Quattro Pro (Windows)	CGMBQWC (disabled in dgdevice catalog)
Borland Quattro Pro (DOS)	CGMBQA* (disabled in dgdevice catalog)
Frame Tech FrameMaker	CGMFRCA* (disabled in dgdevice catalog) CGMFRGA* (disabled in dgdevice catalog) CGMFRMA* (disabled in dgdevice catalog)
Harvard Graphics 2.12 for DOS	CGHHG (disabled in dgdevice catalog)
Harvard Graphics 3.0 for DOS	CGMHG3A* (disabled in dgdevice catalog)
Harvard Graphics for Windows	CGMHGWA* (disabled in dgdevice catalog)
ImageBuilder	CGMIMG (disabled in dgdevice catalog)
Interleaf 5	CGMCILFC* (disabled in dgdevice catalog) CGMGILFG* (disabled in dgdevice catalog) CGMMILFM* (disabled in dgdevice catalog)
Microsoft Word for Windows 6.0	CGMMW6C* (disabled in dgdevice catalog)
Microsoft Word for Windows 2.0	CGMMWWC* (disabled in dgdevice catalog)
Microsoft PowerPoint	CGMMPPA (disabled in dgdevice catalog)
Microsoft Office 97	CGMOF97L or CGMOFML, CGMOF97P, or CGMOFMP
Polaroid CI3000	CI3000 (disabled in dgdevice catalog)
WordPerfect 5.1 for DOS	CGMWPCA (disabled in dgdevice catalog) CGMWPCAP* (disabled in dgdevice catalog) CGMWPGA (disabled in dgdevice catalog) CGMWPGAP* (disabled in dgdevice catalog) CGMWPMA (disabled in dgdevice catalog) CGMWPMAP* (disabled in dgdevice catalog)

Package	Suggested Driver
WordPerfect 5.2 for Windows	CGMWPWA* (disabled in dgdevice catalog)
WordPerfect Presents for DOS	CGMWPCA* (disabled in dgdevice catalog) CGMWPGA* (disabled in dgdevice catalog) CGMWPMA* (disabled in dgdevice catalog)
Zenographics Pixie	CGMPIX (disabled in dgdevice catalog)

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 information for your target software application.

Additional Resources

For full details about using SAS/GRAPH software, see *SAS/GRAPH: Reference*. For further details about using graphics and fonts with SAS under Windows, see "Using TrueType Fonts with SAS/GRAPH Software" on page 765 .

7

Performance Considerations under Windows

Hardware Considerations 245
Overview of Hardware Considerations 245
Processor Speed 246
Memory 246
I/O Subsystem Configuration 246
Windows Features That Optimize Performance 247
Controlling SAS Responsiveness 247
I/O Enhancements for Multiple Processors 248
Memory-Based Libraries 249
SAS Features That Optimize Performance 252
Network Performance Considerations 254
Sasiotest Utility 254
Advanced Performance Tuning Methods 254
Overview of Advanced Performance Tuning Methods 254
Improving Performance of the SORT Procedure 254
Calculating Data Set Size 256
Increasing the Efficiency of Interactive Processing 259

Hardware Considerations

Overview of Hardware Considerations

The following hardware factors might affect SAS performance:

- the processor speed
- the amount of physical memory that is available

- the amount of throughput to disk that is available for I/O
- the graphics adapter.

Not all of these factors applies to your particular configuration or to how you run SAS. Consult your system administrator for details.

See *System Requirements for SAS 9.4 Foundation for Microsoft Windows* and *System Requirements for SAS 9.4 Foundation for Microsoft Windows for x64* for details about memory and hardware considerations.

Processor Speed

In general, a faster processor enables the computer to perform more operations per second. Performance improves when more operations can be performed.

The amount of processor cache that is available also influences performance. More processing cache results in better performance.

- In 32-bit SAS, you must have a PC that contains an Intel Pentium 4 class processor or a processor that is compatible with Intel.
- In x64 64-bit SAS, you must have an Intel64 or AMD64 processor.

Memory

In general, more physical memory results in better performance.

I/O Subsystem Configuration

An application uses I/O for data storage and data access. Therefore, faster I/O results in faster overall performance. Some important 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 configuration can improve I/O performance.

  The type of I/O controller can affect I/O performance. Consult your Storage Administrator and see the following documentation for configuring your storage devices: http://support.sas.com/resources/papers/proceedings15/SAS1501-2015.pdf.

hard drive
: Since SAS is heavily I/O oriented, access time and transfer rate are important to system performance. For best Windows Server performance, I/O throughput rates of at least 100 MB/sec per file system used by SAS are recommended.

  Note: Your storage administrator can configure the I/O subsystem for the best I/O throughput rate for your storage setup.

  Low disk space or a heavily fragmented disk can hinder I/O performance. You should defragment your hard drive regularly to keep I/O performance from degrading.

  See the following documentation for information regard the RAID system: http://support.sas.com/resources/papers/proceedings15/SAS1501-2015.pdf

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.

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 267 .

Optimizing Application Performance under Windows

Under Windows, follow these steps to optimize application performance:

1. Open the Control Panel.
2. Click **System and Security**.
3. Select the **System**.
4. Click **Advanced system settings task**.
5. Select the **Advanced** tab.
6. In the **Performance** box, click **Settings** and then select the **Advanced** tab.
7. To optimize performance of an interactive SAS session, select **Programs**.
8. To optimize performance of a batch SAS session, select **Background services**.
9. 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 processing 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.

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 to support these memory-based libraries.

In Windows operating environments, 32-bit processing uses 2 gigabytes (GB) of physical memory for the operating environment, and the remaining 2 GB of physical memory is 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. Memory 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 in several ways:

- 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

After you have completed the setup for your operating environment, use the MEMLIB and MEMCACHE system options and the MEMLIB option in the LIBNAME statement to access memory-based libraries.

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 and thus cause 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:

- an Intel 64-bit processor or 64-bit processor that is compatible with Intel.
- 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 9.4 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: Windows” on page 645 .
- Set the memory block size for memory-based libraries by using the MEMBLKSZ system option. See “MEMBLKSZ System Option: Windows” on page 642 .

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 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: Windows” on page 644 and the “LIBNAME Statement: Windows” on page 556 .

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 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.

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: Windows" on page 643 . For more information about Scatter Gather I/O, see "SAS Features That Optimize Performance" on page 252 and "SGIO System Option: Windows" on page 683 .

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 Programmer's Guide: Essentials*.

- 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 9.4 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 action 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 action 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 Data Set Options: Reference*.
- If you specify the Scatter-read/Gather-write system option, SGIO, SAS bypasses intermediate buffer transfers when reading or writing data. SAS reads 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. For more information see Achieving Better I/O Throughput Using SGIO in the Microsoft Windows Environment

  Scatter–read / gather–write is active only for SAS I/O opened in INPUT, OUTPUT mode, and UPDATE mode if the access pattern is sequential. 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 about the SGIO system option, see “SGIO System Option: Windows” on page 683 .

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.

Sasiotest Utility

The sasiotest utility measures I/O on a Windows platform. The utility is external to SAS. Sasiotest opens files and reads and writes data in a way that is similar to SAS I/O. For more information, see "Sasiotest Utility" on page 813.

Advanced Performance Tuning Methods

Overview of 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 how SAS is configured on your machine.

Improving Performance of the SORT Procedure

Overview of 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 "SORT Procedure Statement: Windows" on page 529 .

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 SAS Work directory to complete the sort.

The default value of this option is 1G.

TAGSORT Option

The TAGSORT option is useful in single-threaded situations where there might 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 might be much higher.

Choosing a Location for the Sorted File

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 for both single-threaded and multi-threaded sorts. If two or more locations are specified for the UTILLOC system option, the next available 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, ensure that your Work data library has room for this temporary utility file.

The sorted data set replaces the input data set anytime the OUT= option specifies the same name as the IN/DATA= data set. The OVERWRITE option allows for the input data set to be deleted before the output data set has been created , decreasing the peak disk space requirements.

The output data set is populated from the contents of the utility file. The original data set is deleted after the sort is complete if the output data set is replacing the input data set. Before you sort a data set, make sure that you have space for the .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 in the subdirectory that was created for the SAS session within the specified WORK directory.

  ```
  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 and second sorts occurs in the SAS Work subdirectory:

  ```
  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 in the PROC SORT statement, you need space equal to the data set size. If you do not specify the OVERWRITE option, the space that you need is equal to two times the data set

size. For more information about the OVERWRITE option, see the “Sort Procedure” in the *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 that 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, October              Observation Length    40
                     3, 2007 07:41:04
Last Modified        Wednesday, October              Deleted Observations  0
                     10, 2007 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             9.0201B0
Host Created                XP_PRO


                 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 on page 258 , 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 action 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 Windows System Tools with SAS under Windows **263**

Chapter 9
Using OLE in SAS/AF Software under Windows **285**

Chapter 10
Controlling SAS from Another Application Using OLE **315**

Chapter 11
Using Unnamed and Named Pipes under Windows **325**

Chapter 12
Accessing External DLLs from SAS under Windows **339**

Chapter 13
Special Considerations for SAS/AF Programmers under Windows **363**

8

Using Windows System Tools with SAS under Windows

Introduction to Using Windows System Tools with SAS **263**

Event Viewer Application Log **264**
Accessing the Application Log Using Windows 7 264
Viewing a SAS Event 264
Sending Messages to the Application Log Using a User-Written Function 265
Sending Messages to the Application Log Using LOGEVENT.EXE 266

Performance Tools **267**
Why Use a Performance Monitor? 267
Performance Counters and Objects 268
Starting the Windows Performance Monitors 268
SAS Counters in the Performance and System Monitors 268
Selecting SAS Counters to Monitor 269
Examples of Monitoring the DATA Step, PROC SORT, and PROC SQL 270
Examining the Performance between the DATA and PROC SORT Steps 270
Examining a PROC SQL Query 272

Starting SAS as a Windows Service **274**
Overview of Starting SAS as a Windows Service 274
Starting the SAS Service Configuration Utility 274
Creating an Initialization File 275
Installing a SAS Service 282
Starting a SAS Service 284
Removing a SAS Service 284

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 on addition to the SAS Log. Also, informational messages from SAS/CONNECT software can be viewed on the Application Log. For more information, see "Event Viewer Application Log" on page 264 .

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 274 .

Event Viewer Application Log

Accessing the Application Log Using Windows 7

To open the Application Log:

1. Open the Control Panel.
2. Select **System and security** .
3. Select **Administrative tools**.
4. Double-click **Event Viewer**.

   Note: Select **Continue** if you receive a message stating that Windows needs your permission to continue.

5. In the tree view, select **Windows logs** and then click on the left arrow to expand the window logs.
6. Select **Application log**.

An alternate method of starting the Event Viewer is to enter `eventvwr` in the **Search programs and files** dialog box and click **OK**.

Viewing a SAS Event

If a SAS task ends abnormally, information about 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 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 appears 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 8.1 *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 appears 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 displays the message in the **Description** box.

For information about LOGEVENT.EXE, see the documentation for the Windows Resource Kit.

Performance Tools

Why Use a Performance Monitor?

Windows performance monitors are useful for tuning and diagnosing problems in your application or computer system. The monitors include System Monitor under supported Windows operating systems. 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 I/O Read and Write Bytes/Sec for the process counter for the SAS process.

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 per 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.

Starting the Windows Performance Monitors

When you enter **`perfmon`** in the **Search** dialog box from the **Start** menu, you open the Performance window.

Note: To use the performance counters for 32-bit versions of SAS on a Windows x64 system, invoke the 32-bit version of PerfMon after the 32-bit version of SAS has been invoked. To invoke the 32-bit version of PerfMon, access Windows Explorer and open the folder **`c:\Windows\SysWOW64`** that contains the 32-bit applications. In this folder, you see the 32-bit version of PerfMon. You can access the 32-bit versions of SAS performance counters by launchingperfmon.exe or perfmon.msc.

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 8.2 *Procedure for Selecting SAS Counters*

Using Windows
■ Enter `perfmon` in the **Run** dialog box. The **Performance Monitor** opens. ■ Select Performance Monitor window under the Monitoring Tools menu. ■ Select the **+** symbol button and add the counters that you want to monitor. ■ From the **Select counters from computer** dialog box, select the computer associated with the counters. ■ Select **Process** from the list of available counters. ■ From the **Instance of selected object** list select **SAS**. ■ Select **Add>>** to move the counters to the Added Counters dialog box. ■ Select **OK**.

The performance monitor immediately collects and displays information about the counters that you selected.

Multiple SAS counters can be monitored. You can 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. Click on the downward arrow to expand the selection.
4. Add these SAS counters by selecting the counter and then selecting **ADD>>**:
   - Disk ReadFile Bytes Read/Sec
   - Disk WriteFile Bytes Written/Sec
   - Disk SetFilePointer/Sec
5. Click **OK** to close the **Counters** dialog box.
6. Select the Process object.
7. Add these Process counters:
   - %Processor Time
   - %User Time
   - %Privileged Time
8. Click **Done** or **Close**.

Examining the Performance between the DATA and PROC SORT Steps

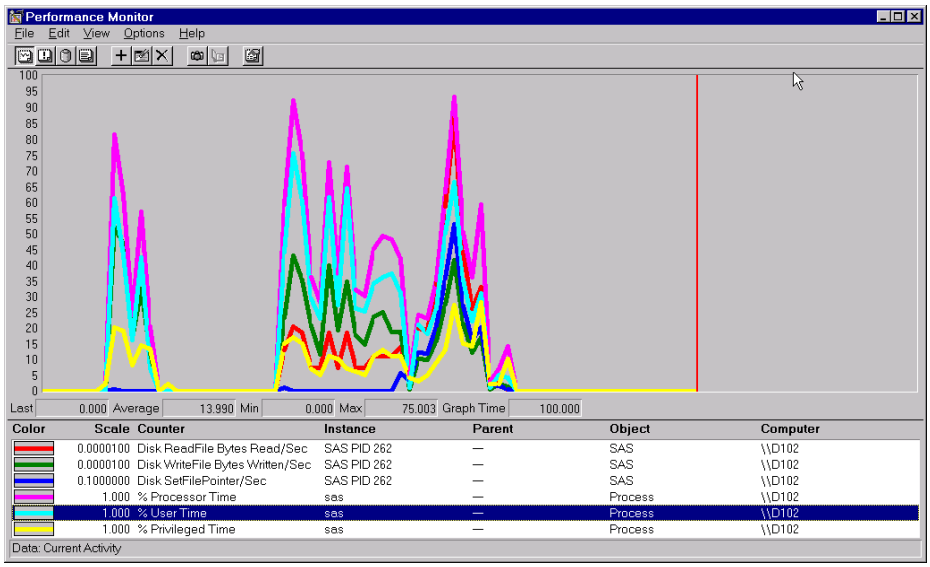
To see the difference in performance between the DATA step and the PROC step, submit this code:

```
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 generates results similar to those results in Figure 8.1 on page 271 . You might have to adjust the scale factor of the different counters.

Figure 8.1 *Performance of the DATA Step and the PROC SORT Step*



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. The output 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 the 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 Figure 8.2 on page 273 .

```
      /* 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 Figure 8.3 on page 273 .

Figure 8.2 *Performance of PROC SQL Query with an Index*

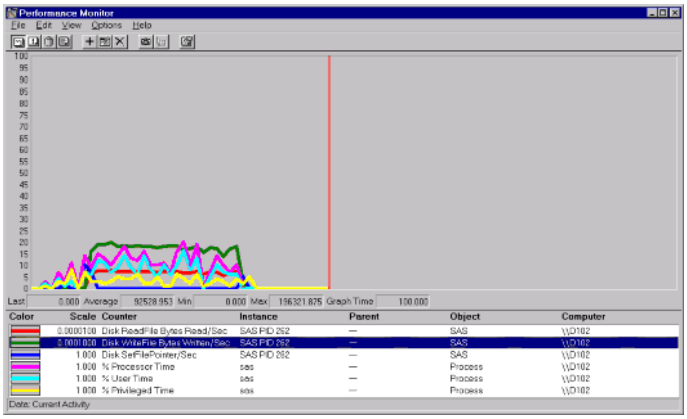
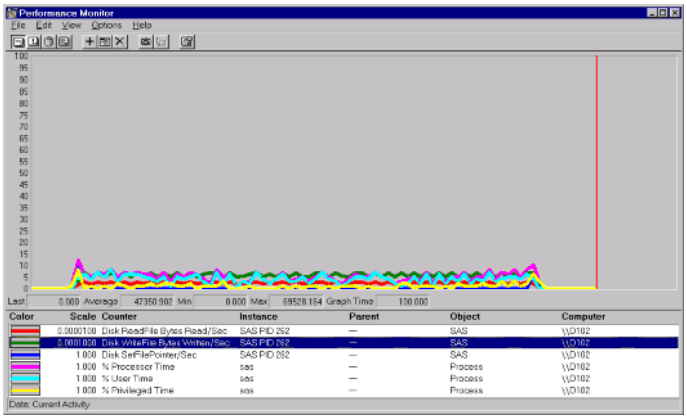


Figure 8.3 *Performance of PROC SQL Query without an Index*



In Figure 8.3 on page 273 , the counters averaged under 10% on the scale, whereas in Figure 8.2 on page 273 , 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 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 the counters by observing the absolute values. The Average value for Disk WriteFile Bytes Written/Sec in Figure 8.2 on page 273 was 92528.953. Contrast this value with the same counter in Figure 8.3 on page 273 , 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 from 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 the Services window in all Windows operating environments. For more information, see "Starting a SAS Service" on page 284 .

If you have multiple SAS configurations, you can create an initialization file for each configuration.

SAS provides the SAS Service Configuration Utility (SAS SCU) to configure the service and install the INI file.

Starting the SAS Service Configuration Utility

To start the SAS SCU, do one of the following:

- Select **Start** ⇨ **Programs** ⇨ **SAS** ⇨ **Utilities** ⇨ **SAS Service Configuration Utility**
- From the SAS SCU directory, type `sasservicemngr.exe`. The default SAS SCU directory is **`c:\Program Files\SASHOME\SASFoundation\ 9.4\core\sasscu`**.

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 Unicode file that does the following:

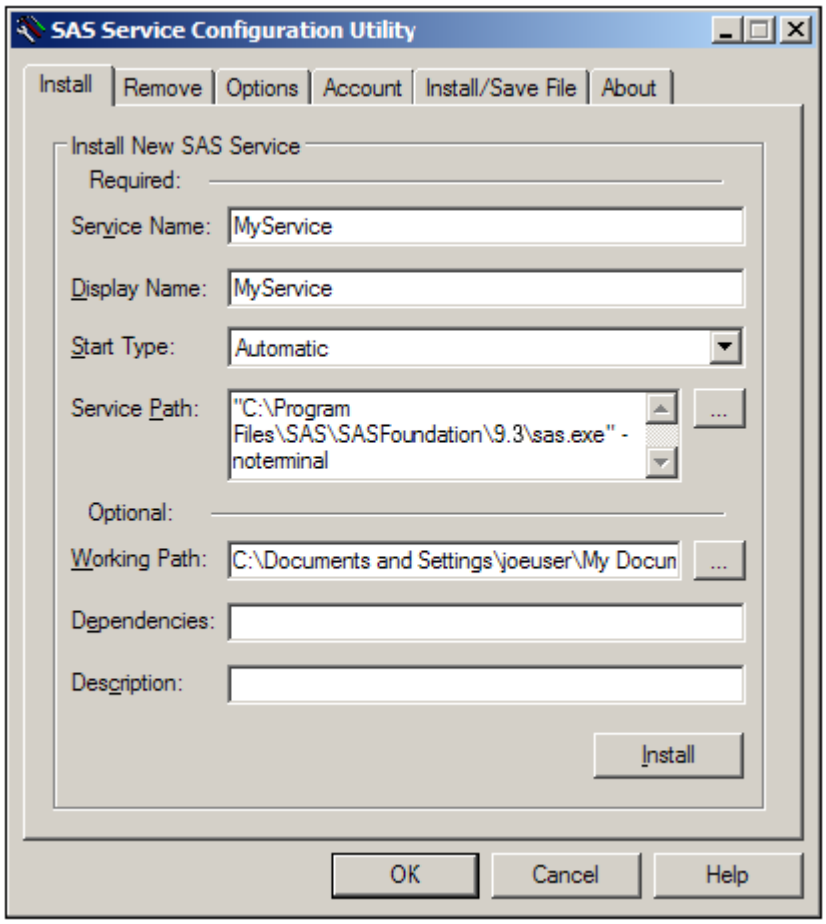
- names the SAS service
- specifies if the service is to start automatically or manually
- defines paths to SAS and SAS working paths
- specifies the level of access that an application has to the SAS service
- names other Windows services that must be started before this service can be started
- defines 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
- specifies whether the user can interact with the SAS desktop.

You create the initialization file either by using the SAS SCU graphical user interface (GUI) or by using a Unicode-capable text editor, such as Notepad. If you use the SAS SCU GUI, you specify only the required values and the SAS SCU creates the INI file for you. If you use a text editor to create the INI file, you must specify the SAS service settings and their values. You must save the file as a Unicode file, not as an ANSI or ASCII-encoded file. Table 8.4 on page 279 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 (SAS SCU) that is shown in the following display to create the INI file. After configuring the settings, click the **Install/Save File** tab to save and install the INI file.

Figure 8.4 SAS Service Configuration Utility



To configure the INI file, select the following tab and modify the appropriate settings:

Install tab

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 field is required. The default is **SASService**.

Display Name
is the name of the service that is displayed to user-interface applications. This field is required. The default is **A SAS Service**.

Start Type
specifies whether the SAS service is started manually, automatically, or is disabled. This field is required. 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 start-up. **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 field is required. For a new

installation, the default path is the SAS installation path, followed by the command **`sas.exe -noterminal`**. If the SAS SCU has been installed previously, the default path is **`c:\Program Files\SASHome\SASFoundation\9.4\`** 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. Under Windows, the user's profile is located at **`c:\Users\`***user ID*`\Documents\My SAS Files\9.4`**. 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 does not be start.

To specify dependencies, type one or more service names separated by the pipe (|) character. An example is NetDDE|NetDDEdsdm.

Description
Enter a description of the service. The description appears in the Windows Services window.

Remove tab

Remove Existing Service
specifies the name of the installed SAS service that you want to remove.

Options tab

Error Control
determines the error severity if the SAS Service fails to start. Select one of the following error controls:

Table 8.3 *Error Controls*

Error Control	Description
Ignore	The error is logged. Start-up operations continue.
Normal	The error is logged and a message is displayed. Start-up operations continue.
Severe	The error is logged and start-up operations continue by using the last successfully installed INI file.
Critical	An attempt to log the error is made. If the start-up operation is using the last known successful INI file, start-up operations fail. If the start-up operation is not using the last known successful INI file, it

Error Control	Description
	attempts 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 `1 (TRUE)` in the INI file. To further configure all access types settings, select the appropriate boxes. For a description of access type settings, see Table 8.4 on page 279 . 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 `1 (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 `1 (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 `1 (TRUE)`.

Account tab

System Account
specifies that the service is shared for all users that log on to 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**, enter the account name in the box. Then enter the password in the **Password** and **Confirm Password** boxes.

Install/Save File tab

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 SAS Service 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 Contents** box.

About tab
displays the copyright information for the SAS Service Configuration Utility.

Creating the Initialization (INI) File Using a Unicode Text Editor

To create a SAS Service INI file by using any Unicode-capable text editor, such as Notepad, 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 8.4 *SAS Service INI File Settings and Default Values*

Setting Name	Required	Explanation	Valid Values	Defaults	Related SAS SCU 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 Pathname=	Yes	Contains the directory path in which the SAS Service INI file is installed, followed by the SAS command to start the service.	The pathname 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 SERVICE_DEMAND_START 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	none	Dependencies

Setting Name	Required	Explanation	Valid Values	Defaults	Related SAS SCU Field
			dependences in quotation marks.		
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.	"Under Windows c:\Users*user ID*"	Working Path
ErrorControl=	Yes	Determines the error severity if the SAS Service fails to start.	SERVICE_ ERROR_ IGNORE SERVICE_ ERROR_ NORMAL SERVICE_ ERROR_ SEVERE SERVICE_ ERROR_ CRITICAL	SERVICE_ ERROR_ NORMAL	Error Control
Interactive=	Yes	Specifies whether the service allows a user to interact with the SAS desktop.	1(TRUE) 0(FALSE)	0(FALSE)	Interactive Process
AccessChgCfg=	Yes	Modifies the SAS Service configuration.	1(TRUE) 0(FALSE)	1(TRUE)	Change Configuration
Access Interrogate=	Yes	Requests that the SAS Service immediately update its current status.	1(TRUE) 0(FALSE)	1(TRUE)	Interrogate
Access PauseCont=	Yes	Pauses and resumes the SAS Service.	1(TRUE) 0(FALSE)	1(TRUE)	Pause Continue
AccessQryCfg=	Yes	Makes queries about the SAS	1(TRUE) 0(FALSE)	1(TRUE)	Query Configuration

Setting Name	Required	Explanation	Valid Values	Defaults	Related SAS SCU Field
		Service configuration.			
AccessQry Status=	Yes	Queries Windows about the status of the SAS Service.	1(TRUE) 0(FALSE)	1(TRUE)	Query Status
AccessStart=	Yes	Starts the SAS Service.	1(TRUE) 0(FALSE)	1(TRUE)	Start Service
AccessStop	Yes	Stops the SAS Service.	1(TRUE) 0(FALSE)	1(TRUE)	Stop Service
AccessUser DefCtrl=	Yes	Specifies a user-defined control code.	1(TRUE) 0(FALSE)	1(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 a text 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 INI file. An encrypted password is necessary only if you specify Password= in an INI file. In comparison, when you create an INI file by using the SAS SCU, you specify a text password. The SAS SCU 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 SAS SCU, from the command prompt, or from within an application.

To install a SAS Service by using the SAS SCU:

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 `sasservicemngr.exe` *`path/filename`*`.ini`. When you install a SAS Service from the command prompt, user messages are disabled.

Note: You can also install a service from the Install Tab for SAS SCU GUI.

When a SAS Service is installed from an application, the command to install the service is `sasservicemngr.exe` *`path/filename`*`.ini`. The following table lists the return codes that can be passed back to the calling application:

Table 8.5 *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_CANCELED_INSTALL	The user canceled the installation.

Numeric Code	Error Code	Description
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 canceled the installation because an INI file was not found.
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.

Numeric Code	Error Code	Description
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 7, Window Server 8, and 10, select **Start** ⇨ **Control Panel** ⇨ **System and security** ⇨ **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 SAS SCU or from the command prompt.

To remove a SAS Service by using the SAS SCU:

1. Open the SAS SCU and click the **Remove** tab.
2. Select the SAS Service from the **Remove Existing Services** box.
3. Click **Remove**.
4. Select **Yes** to confirm the removal of the service from the **Remove Service** dialog box.

To remove a SAS Service from the command prompt, type `sasservicemngr.exe /remove <servicename>`.

9

Using OLE in SAS/AF Software under Windows

About OLE 286

SAS/AF Catalog Compatibility 286

Inserting an OLE Object in a FRAME Entry 287
- Introduction to Inserting an OLE Object in a FRAME Entry 287
- Inserting an OLE Object 287
- Pasting an OLE Object from the Clipboard 288
- Reading an OLE Object from an HSERVICE Entry 289
- Inserting an OLE Object By Dragging It 290

Editing an OLE Object within a FRAME Entry 291

Invoking OLE Verbs 292

Using Linked OLE Objects 293
- Overview of Using Linked OLE Objects 293
- Updating a Linked Object with the Links Dialog Box 294
- Updating a Linked Object Programmatically 295

Converting OLE Objects 295

Automating OLE Objects and Applications 297
- Overview of Automating OLE Objects and Applications 297
- Accessing Array Values Returned by the OLE Automation Server 298
- Using Value Properties 300
- Specifying Optional Parameters in OLE Server Methods 300
- Creating an External OLE Automation Instance 301
- Example: Populating a Microsoft Excel Spreadsheet with SAS Data 303

Using OLE Custom Controls (OCXs) in Your SAS/AF Application 305
- Overview of Using OLE Custom Controls (OCXs) in Your SAS/AF Application 305
- Inserting an OLE Control in a FRAME Entry 305
- Registering OLE Controls 306
- Accessing OLE Control Properties 306
- Interacting with the OLE Control Using SCL Methods 308
- Responding to OLE Control Events 308

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, enabled you to link and embed OLE objects into SAS/AF FRAME entries and SAS/EIS applications. OLE 2.0, which SAS 9.4 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 that 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 about OLE in general, see the documentation for the Windows operating environment. For descriptions of the error messages that you might receive while using OLE features in SAS/AF software, see "Using OLE" on page 759 .

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 librefs to those catalogs in your SAS 9.4 session.

Note: SAS/AF catalogs created in SAS 9.4 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 features that are available in Release 6.08.

HSERVICE entries must be used on the platform in which they were created. Any OLE feature 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.)

Catalogs in a 32-bit architecture cannot be used without migrating to a 64-bit architecture and vice versus. Migration is accomplished through the CPORT and CIMPORT options.

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 inserts 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 that you specify must have been created by one of the applications that you have available to supply OLE objects. For example, if Microsoft Excel is 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 293 .

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. You can also change the **Name** of the object. 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 that 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 that you would like to insert based on the clipboard contents. The object 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 293 .

Note: If you paste data from a temporary source (such as a document that you did not save), SAS cannot 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 **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. You can also change the **Name** of the object. 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 that 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

Inserting an OLE Object into a FRAME Entry

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. Press the mouse button and 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. 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 290 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 that 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 action, 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 be read and not removed). You can

override this default action by using a drag modifier; a key press that indicates 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 onto 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 onto 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 enabling 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

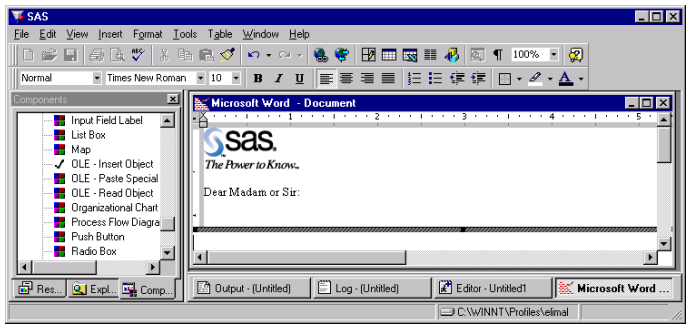
One of the most impressive features of OLE 2.0 is visual editing--the ability to edit an embedded object in-place, without explicitly changing to another application.

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 menus remains the same, but the remainder of the menu bar changes to 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, Figure 9.1 on page 292 shows a SAS/AF FRAME entry with a Microsoft Word object activated.

Figure 9.1 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:

- The HSERVICE entry is automatically updated at the end of a visual editing session only if the object has been saved previously (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.
- 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.
- 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.
- 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 enables you to edit the object. This action 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. (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, 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_ Method" on page 799 .

Using Linked OLE Objects

Overview of 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:

- When you update the source file that an object points to, use the Links dialog box to update the linked object automatically. (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 291). 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 (you must have saved it to disk). If you create a linked object from a temporary data source, SAS is 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

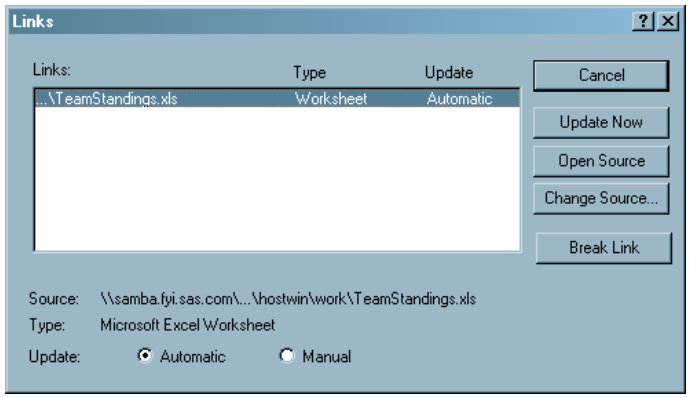
To update the links information with the Links dialog box (shown in Figure 9.2 on page 295):

1. Click on the object with the right mouse button. A pop-up menu appears. The object type is 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');
```

Figure 9.2 Links Dialog Box



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_ Method" on page 805.

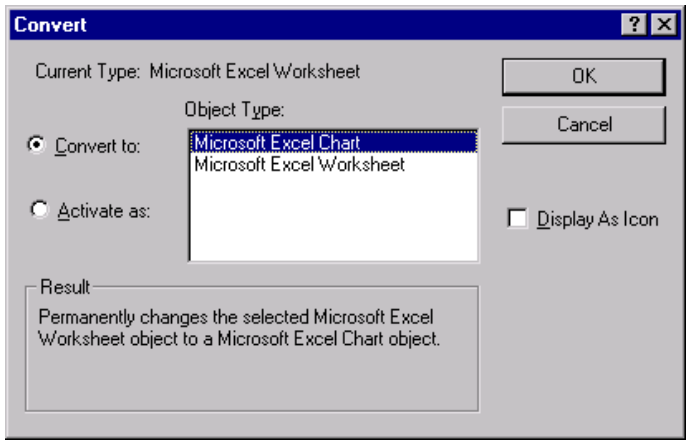
Converting OLE Objects

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 Figure 9.3 on page 296). This dialog box enables you to perform the following actions:

- 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.
- activate the object with a different server application than originally created it, without altering the object type.

Figure 9.3 Convert Dialog Box



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 **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

Overview of Automating OLE Objects and Applications

Some Windows applications provide a scripting language that enables 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 the following processes:

- 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 that you control. OLE automation servers can support multiple types of objects. They can have a unique set of methods and properties. The SCL methods that you can use are listed in Table 9.1 on page 298 and described in detail in "Summary of OLE Class Methods" on page 795 .

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 methods, such as the methods listed in this table.)

Table 9.1 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
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. 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 list box in a Microsoft Excel worksheet and stores the contents of the list box 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', primes);

   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 the following code:

```
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. 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 that 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 that 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. (This action 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 \Windows\SYSTEM32 folder. For more information about 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_ Method" on page 804 .

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.

Table 9.2 *OLE Automation 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 that 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.
Enable 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 9.3 *SCL Code for Populating a Microsoft Excel Spreadsheet*

Action	SCL Code

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:`
	
	`hostcl = loadclass('sashelp.fsp.hauto');`
	`call send(hostcl, '_NEW_', excelobj, 0,`
	`'Excel.Application');`
	`call send(excelobj, '_SET_PROPERTY_', 'Visible', 'True');`
	`return;`

Get the identifier for the current Workbooks property and add a worksheet. Then get the identifier for the new worksheet.	`CREATEWS:`
	`call send(excelobj, '_GET_PROPERTY_', 'Workbooks',`
	`wbsobj);`
	`call send(wbsobj, '_DO_', 'Add' );`
	`call send(excelobj, '_GET_PROPERTY_', 'ActiveSheet',`
	`wsobj);`

Open a SAS data set.	`dsid=open('sasuser.class','i');`
	
	`call set(dsid);`
	`rc=fetch(dsid);`
	`nvar=attrn(dsid, 'NVARS');`
	`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);`

Action	SCL Code

	`   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 that you want to automate. Visual Basic code 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 9.4 *Visual Basic Code Samples and Their SCL Equivalents*

Visual Basic Code	OLE Automation in SCL

Launch Excel and make it visible
`Set excelobj = CreateObject("Excel.Application")`
`excelobj.Visible = True`
	`hostcl = loadclass('sashelp.fsp.hauto');`
	
	`call send ( hostcl,  '_NEW_', excelobj, 0,`
	`   'Excel.Application');`
	`call send (excelobj,'_SET_PROPERTY_',`
	`   'Visible','True');`

Create a new worksheet
`Dim wbsobj, wsobj As Object`
`Set wbsobj = excelobj.Workbooks`
`wbsobj.Add`
`Set wsobj = excelobj.ActiveSheet`
	`call send(excelobj,'_GET_PROPERTY_',`
	`   'Workbooks', wbsobj);`
	`call send(wbsobj,  '_DO_', 'Add');`
	`call send(excelobj,'_GET_PROPERTY_',`
	`   'ActiveSheet', wsobj );`

Set the value of a cell
`wsobj.Cells(row + 1, col).Value`
`=var`
	`r=row+1;`
	`call send(wsobj,'_COMPUTE_', 'Cells', r, col,`
	`   retcell);`
	`call send(retcell,'_SET_PROPERTY_',`
	`   'Value' ,var);`

Close the Excel application object
`excelobj.ActiveWorkbook.Close`
`("False")`
`excelobj.Quit`
	`call send(excelobj,'_GET_PROPERTY_',`
	`'ActiveWorkbook', awbobj);`
	`call send(awbobj, '_DO_', 'Close', 'False');`
	`call send(excelobj,'_DO_', 'Quit');`
	`call send(excelobj,'_TERM_');`

Using OLE Custom Controls (OCXs) in Your SAS/AF Application

Overview of 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 that 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 **OLE - Insert Object** from the Selection List. The Insert Object dialog box appears. 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 appears.

3 Select the **Create Control** radio button to display a list of registered OLE custom controls. If the OLE control that you want to use is not listed here and you have it on your system, you need to register the control. For more information, see "Registering OLE Controls" on page 306 .

4 Select the name of the OCX control that 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 is not 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 305 to invoke the Insert Object dialog box with the list of registered controls.

2 Click **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

Overview of Accessing OLE Control Properties

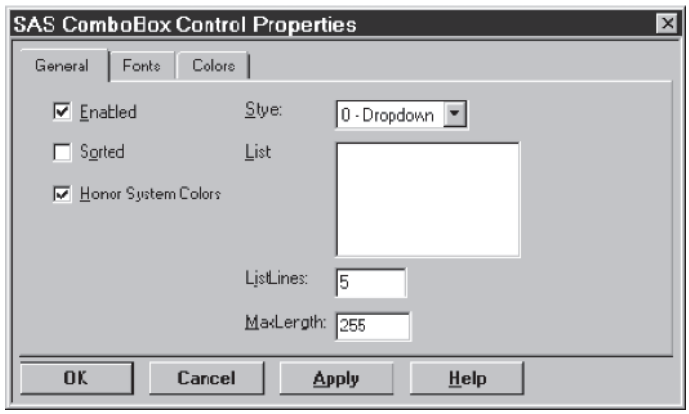
OLE controls have properties that you can set or retrieve using SCL methods. Some controls make some of their properties available through a properties page, which lets you set or retrieve the data interactively.

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. Figure 9.4 on page 307 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, and then choose the cascading menu with the control's name. The Properties verb is available off that cascading menu.

Figure 9.4 *An Example Properties Page*



You can use the properties page to view or change settings for some of the exposed properties.

Note that the control is not active (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 can 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 following information:

- the object label of the OLE control in your SAS/AF FRAME entry
- the name of the property that 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 that 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 contains 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

Overview of 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" on page 309
- "Retrieving Argument Values from Events" on page 310

- "Example: Mapping OLE Control Events to SCL Code" on page 310
- "Example: Subclassing an OLE Custom Control" on page 311
- "Adding an Item to a Combo Box List" on page 311
- "Finding an Item in a Combo Box" on page 312
- "Retrieving the Text Value of the Control" on page 312

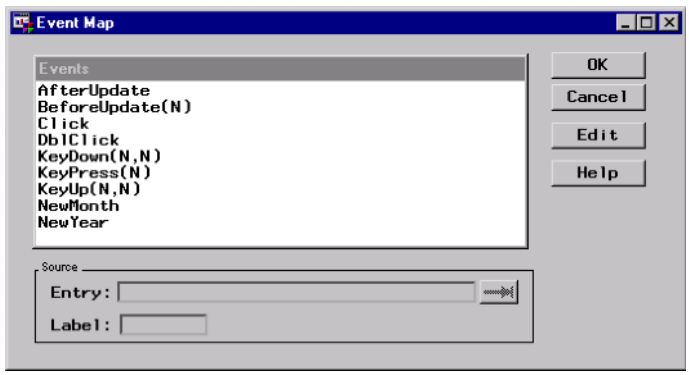
Assigning SCL Code to an OLE Control Event

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 Figure 9.5 on page 309).
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 (outside of the BUILD: SOURCE and BUILD: DISPLAY windows) in order for the event handler to recognize the SCL label. 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.

Figure 9.5 *Event Map Dialog Box*



Note: Many OLE controls include a LostFocus event, which they generate when the control loses window focus. Because of how 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, this action might cause unusual focus behavior.

Retrieving Argument Values from Events

Some OLE control events also include parameters that 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 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. Each piece of code performs 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 write methods 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 that 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 class, 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 (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);
```

10

Controlling SAS from Another Application Using OLE

Introduction to Automating SAS 315
Creating an Instance of SAS 316
Getting Feedback from the SAS Session 316
Examples of Automating SAS with OLE 317
Examples Using Visual Basic 317
Creating a SAS Automation Object 317
Determine Whether the SAS Session Is Busy 317
Toggle the SAS Session between Visible and Invisible 317
Set the Main SAS Window Title of the SAS Session 317
Assign a SAS Library and Run a SAS Procedure 317
End the SAS Session 317
Methods and Properties for Use with a SAS OLE Automation Object 319
Properties for Controlling a SAS Automation Object 319
Dictionary 321
Command Method 321
QueryWindow Method 321
Quit Method 322
Submit Method 322
Top Method 323

Introduction to Automating SAS

SAS can perform as an OLE automation server. You can use an application that can act as an OLE automation controller (such as Visual Basic) to create a SAS session. You can 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 that 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)

Using Visual Studio 2010 vb.net, I had to change this to:
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim OleSAS As Object
        Dim x = 0
        Dim Response
        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")
                x = 1
            End If
        Loop
    End Sub
```

Examples of Automating SAS with OLE

Examples Using Visual Basic

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" on page 318
- "Determine Whether the SAS Session Is Busy " on page 318
- "Toggle the SAS Session between Visible and Invisible" on page 318
- "Set the Main SAS Window Title of the SAS Session" on page 318
- "Assign a SAS Library and Run a SAS Procedure" on page 319
- "End the SAS Session" on page 319

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")
Else
   Response = MsgBox("SAS Session is Idle",
              vbOKOnly, "SAS Session")
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.

```
If OleSAS.Visible = False then
    OleSAS.Visible=True
Else
    OleSAS.Visible=False
End If
```

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 Library and Run a SAS Procedure

This Visual Basic code submits SAS code to the SAS session, assigning a SAS 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 using 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.

Properties for Controlling a SAS Automation Object

The following properties apply to SAS automation objects:

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 that you specify using the Command method. The name that you specify must match the spelling of the window name exactly (though this property is not case sensitive). When the name is set, the window receives subsequent commands

that you specify with the Command method until CommandWindow is changed or set to Null (by specifying ""). If the name is 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 the property is set to `False`, the window that is 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 when 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 box).

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.

Dictionary

Command Method

Invokes a command as if it were 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
```

11

Using Unnamed and Named Pipes under Windows

Overview of Pipes 325

Using Unnamed Pipes 326
- Introduction to Unnamed Pipes 326
- Unnamed Pipe Syntax 327
- Using Redirection Sequences 327
- Unnamed Pipe Example 328

Using Named Pipes 328
- Introduction to Named Pipes 328
- Named Pipe Syntax 329
- Using the CALL RECONNECT Routine 330
- Using Named Pipes in SCL 331
- Named Pipe Examples 331

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 tests. With a one-way pipe, one application can write data only 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 the following numeric file handles:

Table 11.1 *File Handles*

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 327 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 186 .

PIPE
is the device-type keyword that tells SAS that 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 "FILENAME Statement: Windows" on page 542 .

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. Here 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 example demonstrates SAS 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;
```

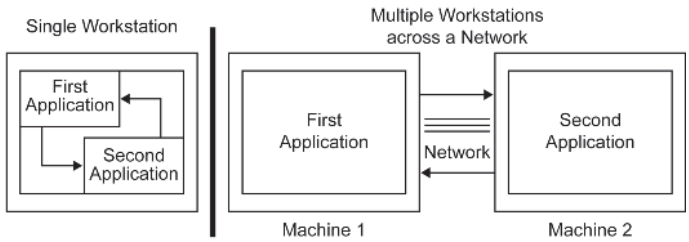
In this example, the PIPE device-type keyword in the FILENAME statement indicates that the fileref STOCKS is an unnamed pipe. The STOCKMKT.EXE reference is the name of the stand-alone program that generates the stock market data. The host-option CONSOLE=MIN indicates that the command prompt window that 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 taskbar). 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 11.1 *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. One or multiple 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 186 .

NAMEPIPE
is the device-type keyword that tells SAS that 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 options 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 336 .

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 that you can use:

Table 11.2 *Nodes When Using SCL Code*

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

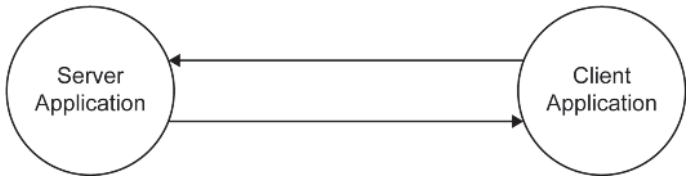
Overview of Named Pipe Examples

The best way to understand named pipes is to examine several 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 11.2 on page 332 .

Figure 11.2 *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:

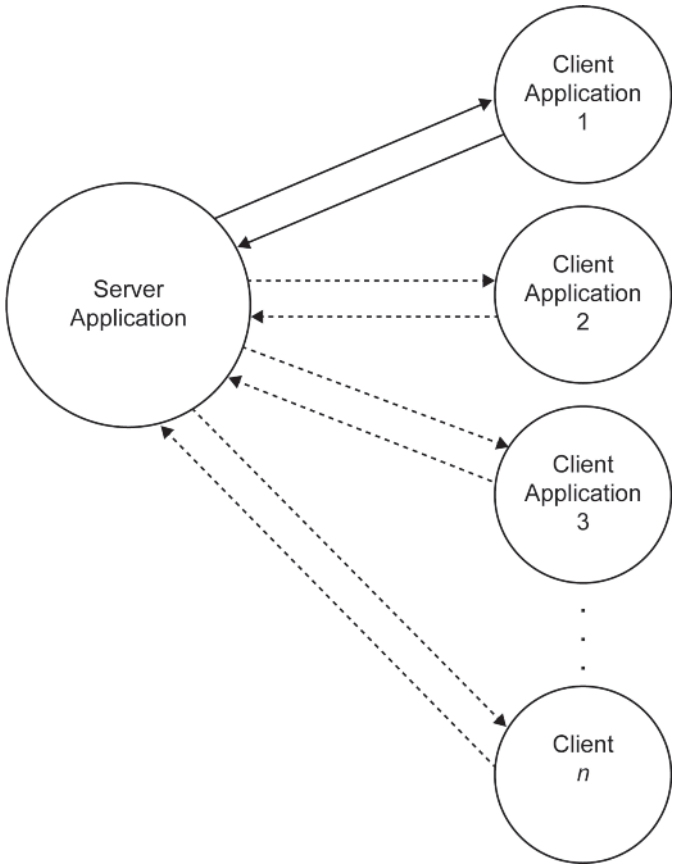
```
filename results namepipe '\\.\pipe\results'
     server retry=60;
proc printto print=results new;
```

```
run;
proc summary data=monthly;
run;
```

One Server Connected to Several Clients

You can choose one server to be connected to several clients. In this case, the named pipe configuration looks like the configuration shown in Figure 11.3 on page 333 .

Figure 11.3 *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 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;
```

12

Accessing External DLLs from SAS under Windows

Overview of Dynamic Link Libraries in SAS **339**

The SASCBTBL Attribute Table **340**
Overview of the SASCBTBL Attribute Table 340
Syntax of the Attribute Table 341
The Importance of the Attribute Table 345

Special Considerations When Using External DLLs **346**
Using PEEKLONG Functions to Access Character String Arguments 346
Accessing External DLLs Efficiently 347
Grouping SAS Variables as Structure Arguments 348
Using Constants and Expressions as Arguments to MODULE 350
Specifying Formats and Informats to Use with MODULE Arguments 351
Understanding MODULE Log Messages 356

Examples **358**
Updating a Character String Argument 358
Passing Arguments by Value 359
Using PEEKCLONG to Access a Returned Pointer 359
Using Structures 360
Invoking a DLL Routine from PROC IML 361

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 that you want to access, including the arguments that it expects and the values that it returns (if any). This attribute file must be in a special format, as described in "The SASCBTBL Attribute Table" on page 340 .
2. Use the FILENAME statement to assign the SASCBTBL fileref to the attribute file that 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 that 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: Windows" on page 487 .

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

Overview of 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. 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 that 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 that you intend to call (using a ROUTINE statement) plus descriptions of each argument associated with the routine (using ARG statements).

Syntax of the Attribute Table

Overview of the 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

Here 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|INT64|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 that you intend to call using the MODULE function. The value for *name* must match the routine name or ordinal that 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 is 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 (the code that calls the routine). Some routines that use Microsofts __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 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 that you are calling is not unique in the attribute table). The MODULE function is described in "MODULE Function: Windows" on page 487 .

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 | INT64 | ULONG | DOUBLE | DBLPTR | CHAR<*n*>
specifies the type of value that the DLL routine returns. This value is 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

INT64
long 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 is 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 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 in which the arguments are 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 attribute is required. Define the arguments in ascending order, starting with the first routine argument (ARG 1).

NUM | CHAR
defines the argument as numeric or character. This attribute is required.

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 whether 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 that 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 num input  byvalue format=ib4.;
arg 2 char output  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 action 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 that 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;
```

Ensure 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 and 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 takes, and the second argument points to a pointer that is 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 this code:

```
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 you need to 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 Functions and CALL Routines: Reference* .

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 that 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 this 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 had this code:

```
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 process 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 are to pass a pointer to a structure. Some parts of the structure might be used as input to the routine. 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 structure 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 data (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 process also requires the use of another routine, GetForegroundWindow, to get the window handle for the window that 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 gathered into a single parameter block.

Here is the output from the log resulting from the PUT statement.

```
LEFT=2 TOP=2 RIGHT=400 BOTTOM=587
```

Using Constants and Expressions as Arguments to MODULE

You can pass any type 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 that you wanted to update is 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. The returned value is lost.

Specifying Formats and Informats to Use with MODULE Arguments

Overview of 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 that 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.

Note: For information about passing character data other than as pointers to character strings, see "$BYVALw. Format" on page 355 .

C Language Formats

Table 12.1 *C Language Formats*

C Type	SAS Format Informat
double	RB8.
float	FLOAT4.
signed into	IB4.
signed short	IB2.
signed long	IB4.
CHAR *	IB4. (32-bit SAS)
char *	IB8 (x64)

C Type	SAS Format Informat
unsigned int	PIB4.
unsigned short	PIB2.
unsigned long	PIB4.
char[*w*]	$CHAR*w*. or $CSTR*w*. (see "$CSTRw. Format" on page 354)

Fortran Language Formats

Table 12.2 *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

Table 12.3 *PL/I Language Formats*

PL/I Type	SAS Format Informat
FIXED BIN(15)	IB2.
FIXED BIN(31)	IB4.

PL/I Type	SAS Format Informat
FLOAT BIN(21)	RB4.
FLOAT BIN(31)	RB8.
CHARACTER(*w*)	$CHAR*w*.

The PL/I descriptions are added here for completeness; this action does not guarantee that you can invoke PL/I routines.

COBOL Language Formats

Table 12.4 *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.

Table 12.5 *COBOL Language Formats (zoned and packed decimal)*

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 usable in conjunction with the corresponding S370F*xxx* informat and format. The S370F elements allow for IBM mainframe-style representations to be read and written in the PC environment.

Table 12.6 *COBOL Language Formats (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

$CSTRw. 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, the following code fragment displays an 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 action 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.

$BYVALw. 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 and informat called $BYVAL*w*. The $BYVAL*w*. format and informat expects a single character and can 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 value is because an ASCII `X` is stored as hexadecimal 58, and this value 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 action occurs because `X` and `Q` values are automatically promoted to ints (which are the same as longs for the sake of this example). 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 12.1 on page 357 to be printed in the log.

Output 12.1 *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
0000000000002C400000000000002E4000000000000030400000000000003140000000000000324000
0000000000038400000000000003940000000000003A400000000000003B400000000000003C400000
0000000000414000000000080414000000000
---ROUTINE changi LOADED AT ADDRESS 886119B8 (PARMLIST AT 886033A0)--- PARM 1
06000000     <CALL-BY-VALUE>
PARM 2 88604720
0E0000000F000000100000001100000012000000180000001900000001A0000001B0000001C000000
2200000023000000240000002500000026000000 2C0000002D0000002E0000002F00000030000000
---VALUES UPON RETURN FROM changi ROUTINE---   PARM 1 06000000
<CALL-BY-VALUE>
PARM 2 88604720
140000001F0000002A0000003500000040000000820000008D00000098000000A3000000AE000000
F0000000FB000000060100001101000 1C0100005E0100006901000074010000 7F0100008A010000
---VALUES UPON RETURN FROM MODULEIN ROUTINE---  NUM PARM 3 885E0AE0
0000000000001840
NUM PARM 4 885E07F0
000000000003440000000000003F400000000000004540000000000804A400000000000005040
0000000004060400000000000A0614000000000000063400000000006064400000000000C06540
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 hexadecimal 2A69, and the ASCII representation of that value ('*i') is in parentheses after the hexadecimal value. The second parameter is likewise printed. The first two arguments print their ASCII equivalents. They are printed because other arguments might contain unreadable binary data.

  The remaining parameters appear with only hexadecimal 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 hexadecimal 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. 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);
```

Here is the proper attribute table:

```
routine lstrcat
  minarg=2
  maxarg=2
  stackpop=called
  module=KERNEL32
  returns=ptr;
  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;
  length charptr $20;
  string1 = 'This is';
  string2 = ' a test!';
  charptr=modulec('lstrcat',string1,string2);
  concatstr = peekclong(charptr,200);
  put concatstr=;
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 "PEEKCLONG Function" in *SAS Functions and CALL Routines: Reference* and "PEEKLONG Function" in *SAS Functions and CALL Routines: Reference*.

Using Structures

"Grouping SAS Variables as Structure Arguments" on page 348 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;
```

Here 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
```

13

Special Considerations for SAS/AF Programmers under Windows

Controlling the Appearance and Behavior of SAS 363

Controlling the Main SAS Window 364
Overview of Controlling the Main SAS Window 364
SAS Options That Control the Main SAS Window 364
SAS Commands That Control the Main SAS Window 365

Accessing External DLLs from SAS 368

Designing, Saving, and Loading Custom Toolbar Controls 369

Invoking SAS/AF Applications Automatically 370

Associating Your Own Logo and Icons with Your SAS/AF Application 370

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 perform the following actions:

- use SAS 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

Overview of Controlling the Main SAS Window

SAS 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 Options That Control the Main SAS Window

The following table lists the system options that provide control over the main SAS window.

Table 13.1 *SAS 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.

Option	Description
ICON	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 that 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 13.2 on page 366 lists the SAS commands that you can use to control the appearance and behavior of the main SAS window.

Table 13.2 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.
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.

Option	Description
DLGSAVE	Invokes the **Save As** dialog box.
DLGSMAIL	Invokes the **Email** 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 toolboxes.
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 email 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 that you attach to an email message.
WHIDECURSOR	Suppress display of the cursor.
WHSBAR	Toggles the horizontal scroll bars on or off.
WINSERT	Toggles the Insert mode on or off.

Option	Description
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.
WPOPUP	Causes the pop-up menu for the active window to appear.
WSCREENTIPS	Toggles the ScreenTips on or off.
WSTATUSLN	Toggles the status bar 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 action lets you access DLLs that you create or purchase; you can even access operating system DLLs.

To access an external DLL, you need the following information:

- 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 that you want to access, including the arguments that it expects and the values that 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 that you created.
3. In a DATA step or SCL code, use MODULE, MODULEN, or MODULEC to invoke the DLL routine. The specific function that 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 13, "Special Considerations for SAS/AF Programmers under Windows," on page 363 .

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 95 .

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 toolbar:

```
-initcmd "AF c=mylib.myapp.myfirst.frame;
  toolload bar mylib.myapp.profile.toolbox"
```

For more information about the INITCMD system option, see *SAS System Options: Reference*.

Associating Your Own Logo and Icons with Your SAS/AF Application

You can substitute your own logo screen and icons in place of those icons provided by SAS.

Note: These procedures involve creating resources for and building your own dynamic link libraries (DLLs). For more information about creating DLLs, see the Microsoft Win32 Software Development Kit.

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 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: Windows" on page 694.

Your logo is displayed 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 that 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: Windows" on page 712 .
2. Use SAS/AF software to create a FRAME entry.
3. Select the buttonStyle attribute of the push button to display an icon. You can select the Large Icon from the iconStyle attributes of the push button to enlarge the icon.
4. Click the ellipses for the value of the icon attribute for the push button to display an icon. The Select Icon window appears. Icon categories are displayed at the top of the window. Click on the down arrow and then select the User Icons category. The user-defined icons from your resource file are displayed. Select an icon for your push button.

PART 3

Features of the SAS Language for Windows

Chapter 14
Data Set Options under Windows 375

Chapter 15
SAS Commands under Windows 379

Chapter 16
SAS Formats under Windows 447

Chapter 17
SAS Functions and CALL Routines under Windows 461

Chapter 18
SAS Informats under Windows 497

Chapter 19
SAS Procedures under Windows 511

Chapter 20
SAS Statements under Windows 535

Chapter 21
SAS System Options under Windows 569

Chapter 22
Length and Precision of Variables under Windows 725

Chapter 23
SAS Macro Facility under Windows 729

14

Data Set Options under Windows

SAS Data Set Options under Windows 375
Dictionary 375
SGIO Data Set Option: Windows 375

SAS Data Set Options under Windows

Data set options specify actions that apply only to a SAS data set. Using data set options enables you to perform these actions:

- rename variables
- select the first or last *n* observations for processing
- drop variables from processing or from the output data set
- specify a password for a data set

Dictionary

SGIO Data Set Option: Windows

Activates the Scatter and Gather I/O feature for a data set.

Valid in: DATA steps and PROC steps

Syntax

SGIO=YES | NO

Required Arguments

YES
specifies that SAS activate the scatter-read / gather-write feature for a SAS data set. The scatter-read / gather-write feature remains active until your SAS session ends.

NO
specifies that SAS not activate the scatter-read/gather-write feature for the SAS data set.

Details

You can specify the SGIO data set option for any SAS I/O file that is referenced within your SAS code. If you want most of your SAS I/O to be processed with SGIO, then specify the SGIO system option and disable SGIO (SGIO=no). SGIO will not be active for those data sets.

Comparisons

The SGIO data set option specifies that SAS process the data set by using SGIO. The SGIO system option specifies that all SAS files are processed by using SGIO.

Examples

Example 1: SGIO Option

You can specify the SGIO data set option for any SAS I/O file that is referenced in a SAS job. This example is a simple case for the DATA step:

```
data mike(sgio=yes);
    input x y z;
    datalines;
1 2 3
run;
```

Example 2: SGIO Option

The following example is more complex:

```
data master(sgio=yes)
   merge daily1(sgio=yes) daily2(sgio=no) daily3(sgio=yes)
... more SAS statements ...
```

```
run;
```

Note: See the following information for more details: SGIO Usage

15

SAS Commands under Windows

SAS Commands under Windows **381**
Overview of SAS Commands under Windows 381
Commands Not Supported in the Windows Operating Environment 381

Dictionary **382**
AUTOSCROLL Command: Windows 382
AWSMAXIMIZE Command: Windows 382
AWSMINIMIZE Command: Windows 383
AWSRESTORE Command: Windows 384
CAPS Command: Windows 384
COLOR Command: Windows 385
COMMAND Command: Windows 386
CUT Command: Windows 387
DLGABOUT Command: Windows 388
DLGCDIR Command: Windows 389
DLGCOLUMNSIZE Command: Windows 389
DLGCOLUMNSORT Command: Windows 390
DLGCONVERT Command: Windows 390
DLGENDR Command: Windows 391
DLGFIND Command: Windows 391
DLGFONT Command: Windows 392
DLGLIB Command: Windows 392
DLGLINKS Command: Windows 393
DLGOPEN Command: Windows 394
DLGPAGESETUP Command: Windows 396
DLGPREF Command: Windows 397
DLGPRT Command: Windows 397
DLGPRTPREVIEW Command: Windows 398
DLGPRTSETUP Command: Windows 399
DLGREPLACE Command: Windows 400
DLGRUN Command: Windows 401
DLGSAVE Command: Windows 401
DLGSMAIL Command: Windows 403
FILE Command: Windows 404
FILEOPEN Command: Windows 406
FILL Command: Windows 407
GSUBMIT Command: Windows 407

HOME Command: Windows 408
ICON Command: Windows 409
INCLUDE Command: Windows 410
PMENU Command: Windows 412
SAVE Command: Windows 413
STORE Command: Windows 413
SUBTOP Command: Windows 414
TOOLCLOSE Command: Windows 415
TOOLEDIT Command: Windows 415
TOOLLARGE Command: Windows 416
TOOLLOAD Command: Windows 417
TOOLSWITCH Command: Windows 418
TOOLTIPS Command: Windows 419
WATTACH Command: Windows 420
WATTENTION Command: Windows 421
WAUTOSAVE Command: Windows 421
WBROWSE Command: Windows 422
WCOPY Command: Windows 423
WCUT Command: Windows 424
WDOCKVIEW Command: Windows 424
WDOCKVIEWMINIMIZE Command: Windows 425
WDOCKVIEWRESIZE Command: Windows 426
WDOCKVIEWRESTORE Command: Windows 426
WEDIT Command: Windows 427
WEMAILFMT Command: Windows 428
WEXITSAVE Command: Windows 429
WFILE Command: Windows 430
WHIDECURSOR Command: Windows 430
WHSBAR Command: Windows 431
WINSERT Command: Windows 432
WMENUPOP Command: Windows 433
WMRU Command: Windows 434
WNAVKEYUNMARK Command: Windows 435
WNEWTITLE Command: Windows 436
WNEXTEDIT Command: Windows 436
WPASTE Command: Windows 437
WPGM Command: Windows 437
WPOPUP Command: Windows 438
WRTFSAVE Command: Windows 439
WSCREENTIPS Command: Windows 439
WSTATUSLN Command: Windows 440
WUNDO Command: Windows 442
WVSBAR Command: Windows 443
WWINDOWBAR Command: Windows 443
X Command: Windows 444
ZOOM Command: Windows 445

SAS Commands under Windows

Overview of 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 to 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 63 .

Commands Not Supported in the Windows Operating Environment

The following SAS commands are not supported under Windows:

PCLEAR	SMARK	WMOVE
PLIST	WDRAG	WSHRINK
SCROLLBAR	WGROW	

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 scroll bars and window sizing bars are an integral part of the graphical user interface.

Dictionary

AUTOSCROLL Command: Windows

Specifies 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 runs.

You can also set scrolling options in the **Preferences** dialog box **Advanced** page.

See Also

- “Setting Session Preferences ” on page 84
- “AUTOSCROLL Command” in the SAS Help and Documentation

AWSMAXIMIZE Command: Windows

Maximizes the main SAS window.

Windows specifics: all

Syntax

AWSMAXIMIZE <ON | OFF>

Without Arguments

toggles the main SAS window between the maximized and the restored state.

Optional Arguments

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 enables you to enlarge the main SAS window to use the complete Windows desktop.

AWSMINIMIZE Command: Windows

Minimizes the main SAS window.

Windows specifics: all

Syntax

AWSMINIMIZE <ON | OFF>

Without Arguments

toggles the main SAS window between the minimized and the restored state.

Optional Arguments

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: Windows

Restores the main SAS window to its previous state.

Windows specifics: all

Syntax

AWSRESTORE <ON | OFF>

Without Arguments

toggles the main SAS window between the maximized and the restored state.

Required Arguments

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: Windows

Specifies whether to write uppercase characters.

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: Windows

Specifies the color and highlighting of selected portions of a window.

Windows specifics: affected window components

Syntax

COLOR *field-type* <*color* | NEXT <*highlight*>>

Required Argument

field-type
specifies the area of the window or the type of text whose color is to be changed.

Optional Arguments

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: Windows

Specifies 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>

Without Arguments

toggles the command line on and off for the active window.

Optional Arguments

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 quotation marks. 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 in which 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 entered with commands that were previously entered.

You must specify the WINDOW or BAR argument in the command before specifying the AUTOCOMPLETE 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 63
- “View Preferences” on page 86
- “DLGPREF Command: Windows” on page 397

CUT Command: Windows

Cuts selected text from a window.

Windows specifics: supported options

Syntax

CUT <LAST | ALL>

Optional Arguments

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 the LAST argument 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 79
- "WCUT Command: Windows" on page 424

DLGABOUT Command: Windows

Opens the About SAS System dialog box.

Windows specifics: all

Syntax

DLGABOUT

Details

To access the **About SAS System** dialog box from the menus, select the **Help** menu and then select `About SAS 9`.

DLGCDIR Command: Windows

Opens the Change Folder dialog box.

Windows specifics: all

Syntax

DLGCDIR

Details

From the **Change Folder** dialog box, you can select a new working folder.

See Also

- “Changing the SAS Current Folder” on page 60
- “SASINITIALFOLDER System Option: Windows” on page 679

DLGCOLUMNSIZE Command: Windows

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.

DLGCOLUMNSORT Command: Windows

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 theSAS Explorer window.

DLGCONVERT Command: Windows

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. This action depends on what the OLE server application supports for that object.

See Also

"Using Linked OLE Objects" on page 293

DLGENDR Command: Windows

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 84

DLGFIND Command: Windows

Opens the Find dialog box.

Windows specifics: all

Syntax

DLGFIND

Details

The **Find** dialog box enables you to search for text strings.

See Also

"DLGREPLACE Command: Windows" on page 400

DLGFONT Command: Windows

Opens the Fonts dialog box.

Windows specifics: all

Syntax

DLGFONT

Details

The **Fonts Selection** dialog box enables you to dynamically change the SAS windowing environment font.

DLGLIB Command: Windows

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.

You can also click **Assign automatically** at start-up to start the library every time you open a new SAS session.

See Also

SAS Help and Documentation for more information about using the SAS Explorer window to manage SAS libraries

DLGLINKS Command: Windows

Opens the Links dialog box.

Windows specifics: all

Syntax

DLGLINKS

Details

The DLGLINKS command opens the **Links** dialog box, enabling you to update a linked object.

If there are no linked objects, the message `There are no linked objects in this window.` appears in the SAS Message Log.

See Also

- "Using Linked OLE Objects" on page 293
- "Using Linked OLE Objects" on page 293

DLGOPEN Command: Windows

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*'>

Without Arguments

opens the **Open** dialog box with the default settings

Optional Arguments

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 filenames that include spaces and single quotation marks. 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.

When you are using the DLGOPEN command in the DM statement, do not use single quotation marks in the LONGFILTER argument. The DM statement requires single quotation marks around the command that it submits. A single quotation mark in the LONGFILTER argument 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 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.

```
dlgopen filter='*.bak *.txt'
```

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
```

loads 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 appears. 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, enabling you to import graphics files into your SAS session. For more information about importing graphics, see "Importing a Graphics File from within a SAS/GRAPH Window" on page 238.

VERIFY

verifies whether the active window contains a File 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"
```

enables 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 menus, select the **File** menu and then select **Open**.

See Also

- “Opening Files” on page 141
- “Setting Session Preferences ” on page 84
- “Importing Graphics from Other Applications” on page 237
- “FILEOPEN Command: Windows” on page 406

DLGPAGESETUP Command: Windows

Opens the Page Setup dialog box.

Windows specifics: all

Syntax

DLGPAGESETUP

Details

The **Page Setup** dialog box enables you to define page attributes such as paper size, source, orientation, and margins.

See Also

“Setting Up the Printed Page” on page 213

DLGPREF Command: Windows

Opens the Preferences dialog box.

Windows specifics: all

Syntax

DLGPREF

Details

The Preferences dialog box enables you to configure your SAS session to accommodate the way that you like to work.

See Also

"Setting Session Preferences " on page 84

DLGPRT Command: Windows

Opens the Print dialog box.

Windows specifics: all

Syntax

DLGPRT <NOSOURCE | ACTIVEBITMAP | SCREENBITMAP | AWSBITMAP | CLIPBITMAP | CLIPTEXT | ALTCMD='*command*' | BITMAPONLY | NODISPLAY | VERIFY>

Without Arguments

prints the active window with the default print settings.

Optional Arguments

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 is printed, using the default settings.

NOSOURCE
prevents the user from specifying a source (application window) from which to be printed.

SCREENBITMAP
suppresses the **Print** dialog box and prints the entire screen as a bitmap.

VERIFY
checks whether the active application window supports text printing (whether the **File** menu contains a `Print` item). If it does not, the **Print** dialog box allows only bitmap printing.

Details

The **Print** dialog box enables you to print the contents of the active window.

See Also

"Printing from within a SAS Window" on page 208

DLGPRTPREVIEW Command: Windows

Invokes the Print Preview window.

Windows specifics: all

Syntax

DLGPRTPREVIEW <VERIFY>

Optional Argument

VERIFY
checks whether the active application window supports printing (that is, whether the File menu contains a `Print` item). If it does not, the **Print Preview** window is not 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 218
- "Printing" on page 208

DLGPRTSETUP Command: Windows

Opens the Print Setup dialog box or programmatically sets printer settings.

Windows specifics: all

Syntax

DLGPRTSETUP <ORIENT=PORTRAIT | LANDSCAPE|> <NODISPLAY>

Optional Arguments

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 enables you to name the printer to which you want to print, specify that you want to use SAS forms to be printed, 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 211
- “Changing the Print Font” on page 212
- “Setting Up the Printed Page” on page 213

DLGREPLACE Command: Windows

Opens the Replace dialog box.

Windows specifics: all

Syntax

DLGREPLACE

Details

The **Replace** dialog box enables you to find a text string and replace it with another text string.

See Also

"DLGFIND Command: Windows" on page 391

DLGRUN Command: Windows

Opens the Run dialog box.

Windows specifics: all

Syntax

DLGRUN

Details

The **Run** dialog box enables you to start another application from within SAS. For example, if you entered excel.exe in the **Command line:** field of the **Run** dialog box, Microsoft Excel would open.

DLGSAVE Command: Windows

Opens the Save As dialog box.

Windows specifics: all

Syntax

DLGSAVE <LONGFILTER="*filters*" | FILTER='*filters*' > <REPLACE><EXPORT> <NOPROMPT> <VERIFY> <ALTCMD='*command*'>

Without Arguments

opens the **Save As** dialog box with the default settings.

Optional Arguments

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 file extensions that include spaces and single quotation marks, 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 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.

When you are using the DLGSAVE command in the DM statement, do not use single quotation marks as part of a LONGFILTER argument. The DM statement requires single quotation marks around the command that it submits. A single quotation mark in a LONGFILTER argument 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 box 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 quotation marks in the filters, where in FILTER='*filters*' you cannot use spaces and 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= arguments as well. For example, the command

```
dlgsave longfilter="*.txt" replace
```

loads the **Files of type:** combo box with only the *.TXT specification (instead of the default file types).

EXPORT
invokes the **Export** dialog box, enabling 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 239.

NOPROMPT
does not prompt the user to replace or append an existing file.

VERIFY
verifies whether the active window contains a File 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 menus, select the **File** menu and then select **Save As** .

See Also

- The Enhanced Editor section, "Saving Files" on page 111
- The Program Editor section, "Saving Files" on page 148

DLGSMAIL Command: Windows

Opens the Send Mail dialog box.

Windows specifics: all

Syntax

DLGSMAIL

Details

The DLGSMAIL command opens the email 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 email dialog box that is compliant with MAPI.

Note: If your email is not set up as an internet Connection, follow the prompts to set up your email

See Also

- "Sending Email Using SAS" on page 65
- "WEMAILFMT Command: Windows" on page 428
- "EMAILDLG System Option: Windows" on page 608

FILE Command: Windows

Writes the contents of the current window to an external file.

Windows specifics: valid options

Syntax

FILE *file-specification* <ENCODING='*encoding-value*'> <*portable-options*> <*host-options*>

Required Argument

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.

Optional Arguments

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): Reference 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 1 megabyte.

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 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

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. If LRECL is not specified, by default SAS reads 32767 bytes at a time from the file.
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 and provides you with these selections:

- Replace
- Append
- Cancel

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

FILEOPEN Command: Windows

Opens the Open dialog box for the Enhanced Editor or opens a file in the Enhanced Editor.

Windows specifics: all

Syntax

FILEOPEN <*"file specification"*>

Required Argument

"*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** appears 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 110

- "DLGOPEN Command: Windows" on page 394

FILL Command: Windows

Specifies the fill character.

Windows specifics: default character

Syntax

FILL *fill-character*

Required Argument

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: Windows

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 are not 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" in the "Using External Files" section in *SAS Companion for Windows*

HOME Command: Windows

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: Windows

Minimizes the active window.

Windows specifics: all

Syntax

ICON <ALL> , <ON> , <OFF>

Without Arguments

specifies that the active window be minimized.

Optional Arguments

ALL
specifies that all windows except the main SAS window be minimized.

ON
specifies that the active window be minimized.

OFF
specifies that the active window be restored to its previous state.

Details

If the window bar is active, the ICON command minimizes windows to the window bar. Otherwise, windows are minimized to the application workspace.

The ICON command (no options) works as a toggle.

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: Windows

Copies the entire content of an external file into the current window.

Windows specifics: valid options

Syntax

INCLUDE *file-specification* <ENCODING='*encoding-value*'> <*portable-options*> <*host-options*>

Required Argument

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.

Optional Arguments

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): Reference 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 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

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 about this option, see "Using the NOTAB Option with DDE" in the "Using Dynamic Data Exchange" section in *SAS Companion for Windows*.

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. If LRECL is not specified, by default SAS reads 32767 bytes at a time from the file.
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 value 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

- "ENCODING= Option Statement" in *SAS National Language Support (NLS): Reference Guide*
- "Referencing External Files" on page 186
- "Using the INCLUDE Command" on page 202
- "Advanced External I/O Techniques" on page 204

PMENU Command: Windows

Toggles the command line in the SAS application windows on and off.

Windows specifics: command behavior

Syntax

PMENU <ON | OFF>

Without Arguments

toggles the command lines on and off.

Optional Arguments

ON
turns the command lines off.

OFF
turns the command lines on.

Details

Under the Windows operating environment, the menus are always enabled. 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: Windows" on page 386
- "WMENUPOP Command: Windows" on page 433

SAVE Command: Windows

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>

Without Arguments

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.

Optional Arguments

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: Windows

Copies selected text or graphics to the Windows clipboard.

Windows specifics: valid options; not supported by the Enhanced Editor

Syntax

STORE <LAST | ALL>

Optional Arguments

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 79

SUBTOP Command: Windows

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*>

Without Arguments

specifies to submit only the top line of the program for processing.

Optional Argument

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 123

TOOLCLOSE Command: Windows

Closes the application toolbar or toolbox.

Windows specifics: all

Syntax

TOOLCLOSE

Details

Use the TOOLCLOSE command to close the toolbar or toolbox.

TOOLEDIT Command: Windows

Opens the Customize Tools dialog box.

Windows specifics: all

Syntax

TOOLEDIT <*library.catalog.entry*>

Without Arguments

edits the currently loaded set of tools.

Optional Argument

library.catalog.entry
specifies the TOOLBOX entry that 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 95

TOOLLARGE Command: Windows

Toggles the size of the toolbar or toolbox buttons.

Windows specifics: all

Syntax

TOOLLARGE <ON | OFF>

Without Arguments

toggles the size of the toolbar or toolbox buttons between large and normal.

Optional Arguments

ON
sets the size of the toolbar or toolbox buttons too 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 100

TOOLLOAD Command: Windows

Loads a specific toolbox.

Windows specifics: all

Syntax

TOOLLOAD <WINDOW> <BOX | BAR> <*libref.catalog.member*>

Without 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.

Optional Arguments

WINDOW
associates the toolbox entry that 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 reverts 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 example, the Explorer window toolset is named Sasuser.Profile.Explorer. If the WINDOW option is not specified, the toolset definition persists 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 99

TOOLSWITCH Command: Windows

Toggles the tool switching feature on and off.

Windows specifics: all

Syntax

TOOLSWITCH ON | OFF

Required Arguments

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 enables you to switch between a toolbar defined for the active window and the SAS default toolbar.

See Also

- “Setting Session Preferences ” on page 84
- “TOOLEDIT Command: Windows” on page 415

TOOLTIPS Command: Windows

Toggles the Tooltips feature.

Windows specifics: all

Syntax

TOOLTIPS <ON | OFF>

Without Arguments

toggles the Tooltips feature on and off.

Optional Arguments

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 bar, the window bar, and tabs in the main SAS window.

See Also

"WSCREENTIPS Command: Windows" on page 439

WATTACH Command: Windows

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>

Without Arguments

toggles the attach mode on and off

Optional Arguments

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 Email Using SAS” on page 65
- “WEMAILFMT Command: Windows” on page 428
- “Setting Session Preferences ” on page 84

WATTENTION Command: Windows

Displays the Tasking Manager window, which enables you to select which SAS process to terminate.

Windows specifics: all

Syntax

WATTENTION

Details

The WATTENTION command enables you to select a SAS process to terminate. This action is the equivalent of pressing Ctrl + Break.

WAUTOSAVE Command: Windows

Controls how often SAS automatically saves work in the SAS editor windows.

Windows specifics: all

Syntax

WAUTOSAVE <<ON | OFF> <INTERVAL=*minutes*>>

Without Arguments

turns the autosave feature on and resets the autosave timer (so that work will automatically be saved after the defined time interval).

Optional Arguments

ON | OFF
specifies to turn the autosave feature on or off.

INTERVAL=*minutes*
saves 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 84
- “AUTOSAVELOC= System Option” in *SAS System Options: Reference*

WBROWSE Command: Windows

Opens the web browser specified in the preferences dialog box.

Windows specifics: all

Syntax

WBROWSE <*"URL"*>

Without Arguments

invokes the preferred web browser as defined in the **Preferences** dialog box **Web** page.

Optional Argument

URL
specifies a URL (Uniform Resource Locator) that 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 (Support.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 84

WCOPY Command: Windows

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 executes the STORE command.

See Also

"STORE Command: Windows" on page 413

WCUT Command: Windows

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 executes the CUT command.

See Also

"CUT Command: Windows" on page 387

WDOCKVIEW Command: Windows

Toggles the Docking View on and off.

Windows specifics: all

Syntax

WDOCKVIEW <ON | OFF>

Without Arguments

toggles the Docking View on and off.

Optional Arguments

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, are displayed 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 57
- “Setting Session Preferences ” on page 84

WDOCKVIEWMINIMIZE Command: Windows

Minimizes the Docking View window.

Windows specifics: all

Syntax

WDOCKVIEWMINIMIZE

Details

WDOCKVIEWMINIMIZE minimizes the Docking View window.

See Also

- “WDOCKVIEWRESTORE Command: Windows” on page 426
- “Using the Docking View ” on page 57

WDOCKVIEWRESIZE Command: Windows

Starts Resize mode for moving the Docking View split bar.

Windows specifics: all

Syntax

WDOCKVIEWRESIZE

Comparisons

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 entering Alt + W + S or by selecting **Window** ⇨ **Size Docking View**

WDOCKVIEWRESTORE Command: Windows

Restores the Docking View window from the taskbar.

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: Windows” on page 425
- “Using the Docking View ” on page 57

WEDIT Command: Windows

Opens an Enhanced Editor window and also enables or disables the Enhanced Editor.

Windows specifics: all

Syntax

WEDIT <*"filename"*> <USE | NOUSE>

Without Arguments

opens an Enhanced Editor window.

Optional Arguments

"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 84
- Chapter 3, "Using the SAS Editors under Windows," on page 107

WEMAILFMT Command: Windows

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

Required Arguments

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 Email Using SAS” on page 65
- “WATTACH Command: Windows” on page 420
- “DLGSMAIL Command: Windows” on page 403

WEXITSAVE Command: Windows

Toggles saving your settings when you exit SAS.

Windows specifics: all

Syntax

WEXITSAVE <ON | OFF>

Without Arguments

toggles the saving of your settings when you exit SAS.

Optional Arguments

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 84

WFILE Command: Windows

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 111
- Program Editor section "Saving Files" on page 148

WHIDECURSOR Command: Windows

Suppresses the display of the cursor in SAS windows that do not allow text input.

Windows specifics: all

Syntax

WHIDECURSOR <ON | OFF>

Without Arguments

toggles between hiding and displaying the cursor.

Optional Arguments

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 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 84

WHSBAR Command: Windows

Toggles the horizontal scroll bars on and off.

Windows specifics: all

Syntax

WHSBAR <ON | OFF>

Without Arguments

toggles the horizontal scroll bars on and off.

Optional Arguments

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 84

WINSERT Command: Windows

Toggles Insert mode on and off.

Windows specifics: all

Syntax

WINSERT <ON | OFF>

Without Arguments

toggles the Insert mode on and off.

Optional Arguments

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 84

WMENUPOP Command: Windows

Toggles the pop-up menus in the SAS application windows on and off.

Windows specifics: all

Syntax

WMENUPOP <ON | OFF>

Without Arguments

toggles the pop-up menus on and off.

Optional Arguments

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 technique can be useful when developing SAS/AF applications in which you want to restrict the actions of the end user.

See Also

- "AWSMENU System Option: Windows" on page 597
- "PMENU Command: Windows" on page 412
- "WPOPUP Command: Windows" on page 438

WMRU Command: Windows

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>

Without Arguments

toggles the file list on and off.

Optional Arguments

ON
turns the file list on.

NUM=*number-of-filenames*
specifies to maintain *number-of-filenames* filenames in the list.

Default The last number specified for the most recently used files

Range an integer, 1–30

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 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 84

WNAVKEYUNMARK Command: Windows

Toggles the setting for enabling unmarking of text using navigational keys.

Windows specifics: all

Syntax

WNAVKEYUNMARK <ON | OFF>

Without Arguments

toggles the **Enable unmarking with navigation keys** setting on and off.

Optional Arguments

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 84

WNEWTITLE Command: Windows

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: Windows

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: Windows" on page 437

WPASTE Command: Windows

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 executes as if you selected PASTE from the menu. If there is no PASTE item, WPASTE executes the PASTE command.

See Also

"PASTE Command" in the SAS Help and Documentation

WPGM Command: Windows

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: Windows" on page 436

WPOPUP Command: Windows

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: Windows" on page 433

WRTFSAVE Command: Windows

Saves the contents of the current text window to an RTF file.

Windows specifics: all

Syntax

WRTFSAVE "*filename*" <NOPROMPT>

Required Argument

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.

Optional Argument

NOPROMPT
specifies that if a file with the same filename already exists, that file is 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.

Note: The WRTFSAVE command does not work with the enhanced editor.

WSCREENTIPS Command: Windows

Toggles the ScreenTips on and off.

Windows specifics: all

Syntax

WSCREENTIPS <ON | OFF>

Without Arguments

toggles the ScreenTips on and off.

Optional Arguments

ON
displays helpful cues for the status bar, 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 bar, 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 bar, 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: Windows” on page 419
- “Setting Session Preferences ” on page 84

WSTATUSLN Command: Windows

Toggles the status bar on and off, and specifies the area proportions.

Windows specifics: all

Syntax

WSTATUSLN <ON | OFF> <ALL | MSG*percent-msg* | CDIR *percent-cdir* <CURPOS> >

Without Arguments

toggles the status bar on and off. The first argument is optional, but if you specify it, you must include it before the second group of options

Optional Arguments

ON
displays the status bar in its most recent active state. If the status bar includes a message area, the message lines in the SAS application windows are disabled. ON is the default setting.

OFF
turns off the status bar. This action 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 bar on if it is currently off.

ALL
includes both the message area and the current folder areas on the status bar. If you do not specify the MSG and CDIR options with percentage values, the status bar proportions revert to the most recent settings. ALL is the default setting.

MSG<=*percent-msg*>
includes the message area as part of the status bar. If you specify this option without the CDIR option, the message area occupies the entire status bar. 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 bar. If you specify this option without the MSG option, the current folder area occupies the entire status bar. 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 bar when the Enhanced Editor is the active window.

Details

The WSTATUSLN command specifies whether the status bar of the active window is on or off and specifies the proportions of the status bar that the message area and the current folder area occupy. You can also toggle the status bar in the **Preferences** dialog box **View** page.

Example

To display a status bar 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 84

WUNDO Command: Windows

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 executes as if you selected **UNDO** from the menu. If there is no undo item, WUNDO executes the UNDO command. Some windows might not have an undo command.

WVSBAR Command: Windows

Toggles the vertical scroll bars on and off.

Windows specifics: all

Syntax

WVSBAR <ON | OFF>

Without Arguments

toggles the vertical scroll bars on and off.

Optional Arguments

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 84

WWINDOWBAR Command: Windows

Toggles the window bar on and off.

Windows specifics: all

Syntax

WWINDOWBAR <ON | OFF>

Without Arguments

toggles the window bar on or off.

Optional Arguments

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 is displayed at the bottom of the main SAS window just above the status bar. SAS windows are minimized 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 are minimized too 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 89
- "Setting Session Preferences " on page 84

X Command: Windows

Enters operating environment mode or enables you to submit a Windows command without ending your SAS session.

Windows specifics: valid values for *command* syntax

Syntax

X <*'command '*> ;

Without Arguments

open a DOS command window.

Optional Argument

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 45
- "XCMD System Option: Windows" on page 719
- "XSYNC System Option: Windows" on page 721
- "XWAIT System Option: Windows" on page 722

ZOOM Command: Windows

Maximizes the active window.

Windows specifics: all

Syntax

ZOOM <ON | OFF>

Optional Arguments

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 that you select is 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

16

SAS Formats under Windows

SAS Formats under Windows **447**
Writing Binary Data **448**
Accessing User-Written Formats from Releases Earlier Than SAS 9.4 ..... **449**
Dictionary **449**
HEXw. Format: Windows 449
$HEXw. Format: Windows 450
IBw.d Format: Windows 451
PDw.d Format: Windows 453
PIBw.d Format: Windows 455
RBw.d Format: Windows 457
ZDw.d Format: Windows 458

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 Formats and Informats: Reference*. 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 448 .

If you have formats that you created for use in earlier releases of SAS, see "Accessing User-Written Formats from Releases Earlier Than SAS 9.4" on page 449 for information about how to convert those formats for use with SAS 9.4.

TIP User-defined format names cannot end in a number. For more information, see "User-Defined Formats" in *SAS Formats and Informats: Reference* and "Words and Names" in *SAS Programmer's Guide: Essentials*.

Writing Binary Data

IBM mainframes, Hewlett Packard 9000, and most other UNIX systems store bytes in one order, called big-endian. Those that are based on Intel, 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 is 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 enable 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 that 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.

Accessing User-Written Formats from Releases Earlier Than SAS 9.4

If you are using 64-bit SAS for Windows, you must use PROC CPORT and CIMPORT to convert. For more information about PROC CPORT and CIMPORT, see *Base SAS Procedures Guide*.

If you are using 32-bit SAS for Windows, and you want to migrate a SAS library, PROC MIGRATE is the recommended method.

Note: User-defined formats are stored as catalog entries.

Dictionary

HEXw. Format: Windows

Converts real binary (floating-point) values to hexadecimal values.

Category:	Numeric
Alignment:	Left
Windows specifics:	native floating–point representation
See:	"HEXw. Format" in *SAS Formats and Informats: Reference*

Syntax

HEX*w*.

Required Argument

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.

Default 8

Range 1–16

Example

The following example uses the input value of 123.

```
data _null_;
x=123;
put x=hex8.;
run;
```

```
x=0000007B
```

See Also

Formats:

- "$HEXw. Format: Windows" on page 450

Informats:

- "HEXw. Informat: Windows" on page 500

$HEXw. Format: Windows

Converts character values to hexadecimal values.

Category:	Character
Alignment:	Left
Windows specifics:	ASCII character–encoding system
See:	$HEXw.

Syntax

$HEX*w*.

Required Argument

w

specifies the width of the output field.

Default 2

Range 1–32767

Details

The $HEX*w*. format is like the HEX*w*. format in that it converts a character value to hexadecimal notation, and each byte requiring two columns. Under Windows, the $HEX*w*. format produces hexadecimal representations of ASCII codes for characters.

Example

The following example uses the input value of 123.

```
data  _null_;
x='123';
put x=$hex8.;
run;
```

```
x=313233
```

See Also

Formats:

- “HEXw. Format: Windows” on page 449

Informats:

- “$HEXw. Informat: Windows” on page 501

IBw.d Format: Windows

Writes values in integer binary (fixed-point) format.

Category: Numeric

Alignment: Left

Windows specifics: native floating-point representation

See: "IBw.d Format" in *SAS Formats and Informats: Reference*

Syntax

IB*w.d*

Required Arguments

w

specifies the width of the output field in bytes (not digits).

Default 4

Range 1–8

d

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.

Range 0–10

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 the Intel developer website.

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 value that you specify*w*.

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 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, IB8.);
   put y = hex16.;
run;
```

SAS returns the hexadecimal representation of 2147483647

```
 y=FFFFFFFFFFFFFF7F
```

See Also

Informats:

- “IBw.d Informat: Windows” on page 502
- “Writing Binary Data” on page 448

PDw.d Format: Windows

Writes values in packed decimal data.

Category:	Numeric
Alignment:	Left
Windows specifics:	How the values are interpreted as negative or positive
See:	“PDw.d Format” in *SAS Formats and Informats: Reference*

Syntax

PD*w*.*d*

Required Arguments

w

specifies the width of the output field in bytes (not digits).

Default 1

Range 1–16

d

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.

Range 1–31

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 99
```

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 = 1e308;
   y = put(x, PD16.2);
   put y = hex16.;
run;
```

the result is

```
y=0099999999999999
```

See Also

Informats:

- "PDw.d Informat: Windows" on page 504
- "Writing Binary Data" on page 448

PIBw.d Format: Windows

Writes values in positive integer-binary (fixed-point) format.

Category:	Numeric
Alignment:	Left
Windows specifics:	native byte-swapped integers
See:	PIBw.d

Syntax

PIB*w.d*

Required Arguments

w

specifies the width of the output field in bytes (not digits).

Default 1

Range 1–18

d

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.

Range 0–10

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 the Intel developer website.

Example: 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=0000E8890423C78A
```

See Also

- "Byte Ordering for Integer Binary Data on Big Endian and Little Endian Platforms" in *SAS Formats and Informats: Reference*
- "Byte Ordering for Integer Binary Data on Big Endian and Little Endian Platforms" in *SAS Formats and Informats: Reference*

Informats:

- "PIBw.d Informat: Windows" on page 505
- "Writing Binary Data" on page 448

RBw.d Format: Windows

Writes values real-binary (floating-point) format.

Category:	Numeric
Alignment:	Left
Default:	4
Ranges:	2–8, 0–10
Windows specifics:	native floating–point representation
See:	"RBw.d Format" in *SAS Formats and Informats: Reference*

Syntax

RB*w.d*

Required Arguments

w

specifies the width of the output field.

Default 4

Range 2–8

d

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.

Range 0–10

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 the Intel developer website.

Example: 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, RB8.2);
   put y = hex16.;
run;
```

the result is

```
y=0000000000D1FFFF
```

See Also

- "Byte Ordering for Integer Binary Data on Big Endian and Little Endian Platforms" in *SAS Formats and Informats: Reference*

Informats:

- "Byte Ordering for Integer Binary Data on Big Endian and Little Endian Platforms" in *SAS Formats and Informats: Reference*
- "RBw.d Informat: Windows" on page 506
- "Writing Binary Data" on page 448

ZDw.d Format: Windows

Writes zoned decimal data.

Category:	Numeric
Alignment:	Left
Windows specifics:	Last byte includes the sign.
See:	"ZDw.d Format" in *SAS Formats and Informats: Reference*

Syntax

ZD*w.d*

Required Arguments

w

specifies the number of bytes (not the number of digits).

Default 1

Range 1–32

d

specifies the number of digits to the right of the decimal point in the numeric value.

Range 1–10

Details

The ZD*w.d* format writes zoned decimal data. This method 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.

Table 16.1 *Last Byte Conversions*

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

Digit	ASCII Character	Digit	ASCII Character
9	I	-9	R

Example: 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

"ZDw.d Informat: Windows" on page 507

17

SAS Functions and CALL Routines under Windows

SAS Functions and CALL Routines under Windows ... **462**
Overview of SAS Functions and CALL Routines under Windows ... 462

Dictionary ... **462**
BYTE Function: Windows ... 462
CALL SOUND Routine: Windows ... 463
CALL SYSTEM Routine: Windows ... 464
COLLATE Function: Windows ... 466
DINFO Function: Windows ... 468
DOPEN Function: Windows ... 469
DOPTNAME Function: Windows ... 470
DOPTNUM Function: Windows ... 471
FDELETE Function: Windows ... 472
FEXIST Function: Windows ... 472
FILEEXIST Function: Windows ... 473
FILENAME Function: Windows ... 474
FILEREF Function: Windows ... 475
FINFO Function: Windows ... 476
FOPTNAME Function: Windows ... 480
FOPTNUM Function: Windows ... 482
LIBNAME Function: Windows ... 483
MCIPISLP Function: Windows ... 484
MCIPISTR Function: Windows ... 486
MODULE Function: Windows ... 487
PEEKLONG Function: Windows ... 489
RANK Function: Windows ... 490
SLEEP Function: Windows ... 491
TRANSLATE Function: Windows ... 493
WAKEUP Function: Windows ... 493

SAS Functions and CALL Routines under Windows

Overview of 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 Functions and CALL Routines: Reference*.

Dictionary

BYTE Function: Windows

Returns one character in the ASCII collating sequence.

Category: Character

Windows specifics: Uses the ASCII code sequence

See: "BYTE Function" in *SAS Functions and CALL Routines: Reference*

Syntax

BYTE(*n*)

Required Argument

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`) can 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: Windows

Generates a sound with a specific frequency and duration.

Category: Special

Windows specifics: all

Syntax

CALL SOUND(*frequency*,*duration*);

Required Arguments

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: 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: Windows

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 Functions and CALL Routines: Reference*

Syntax

CALL SYSTEM(*command*);

Required Argument

command
can be any of the following:

- an operating system command enclosed in quotation marks or the name of a Windows application that is enclosed in quotation marks.
- 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 enter `exit` to return to your SAS session.

Note: The CALL SYSTEM function is not available if SAS is started with NOXCMD.

Comparisons

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;
   datalines;
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: Windows” on page 444
- “XSYNC System Option: Windows” on page 721
- “XWAIT System Option: Windows” on page 722

COLLATE Function: Windows

Returns an ASCII collating sequence character string.

Category: Character

Windows specifics: Uses the ASCII code sequence

See: “COLLATE Function” in *SAS Functions and CALL Routines: Reference*

Syntax

COLLATE (*start-position*< ,*end-position* >)|(*start-position*< ,,*length* >)

Required Argument

start-position
specifies the numeric position in the collating sequence of the first character to be returned.

Optional Arguments

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, and the character has 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`) can 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: Windows

Returns information about a directory.

Category: External Files

Windows specifics: directory pathname is the only information available

See: "DINFO Function" in *SAS Functions and CALL Routines: Reference*

Syntax

DINFO(*directory-id*, *info-item*)

Required Arguments

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: 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;
```

Example Code 17.1 *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: Windows" on page 469

DOPEN Function: Windows

Opens a directory and returns a directory identifier value.

Category: External Files

See: "DOPEN Function" in *SAS Functions and CALL Routines: Reference*
"FILENAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

DOPEN(*"fileref"*)

Required Argument

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: Windows

Returns the name of a directory information item.

Category: External Files

Windows specifics: directory is the only item available

See: "DOPTNAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

DOPTNAME(*directory-id*, *nval*)

Required Arguments

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: Obtaining Directory Information" on page 468.

DOPTNUM Function: Windows

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 Functions and CALL Routines: Reference*

Syntax

DOPTNUM(*directory-id*)

Required Argument

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 returns a value of 1.

Example

For an example of the DOPTNUM function, see "Example: Obtaining Directory Information" on page 468 .

FDELETE Function: Windows

Deletes an external file or an empty directory.

Category: External Files

See: "FDELETE Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FDELETE("*fileref*")

Required Argument

fileref
specifies the fileref that is assigned to the external file or directory. Note that 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 a file. If the function is used within a DATA step, the fileref must be enclosed in quotation marks. If the function is used in a macro, the fileref must not be enclosed in quotation marks.

Details

FDELETE returns 0 if the operation was successful, and a nonzero number if it was not successful.

FEXIST Function: Windows

Verifies the existence of an external file by its fileref.

Category: External Files

See: "FEXIST Function" in *SAS Functions and CALL Routines: Reference*
"FILENAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FEXIST("*fileref*")

Required Argument

fileref
specifies the fileref that is assigned to an external file. Under Windows, *fileref* can also be an environment variable. If the function is used in a DATA step, the fileref or the environment variable that you specify must be enclosed in quotation marks. If the function is used in a macro, then the fileref must not be enclosed in quotation marks.

Details

FEXIST returns a value of 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: Obtaining Directory Information" on page 468.

FILEEXIST Function: Windows

Verifies the existence of an external file by its physical name.

Category: External Files

Restriction: If the SAS session in which you are specifying the FILEEXIST function is in a locked-down state, and the pathname specified in the function has not been added to the lockdown path list, then the function fails and a file access error related to the locked-down data is not generated in the SAS log unless you specify the SYSMSG function.

See: "FILEEXIST Function" in *SAS Functions and CALL Routines: Reference*
"FILENAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FILEEXIST("*filename*")

Required Argument

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 be a complete path, and an environment variable. The filename or environment variable that you specify must be enclosed in quotation marks.

Note The FILEEXIST function can also verify a directory's existence.

Details

FILEEXIST returns 1 if the external file exists and 0 if the external file does not exist.

FILENAME Function: Windows

Assigns or deassigns a fileref for an external file, directory, or output device.

Category: External Files

Restriction: If the SAS session in which you are specifying the FILEEXIST function is in a locked-down state, and the pathname specified in the function has not been added to the lockdown path list, then the function fails and a file access error related to the locked-down data is not generated in the SAS log unless you specify the SYSMSG function.

Windows specifics: SAS 9.4M8The fully-qualified-pathname cannot exceed 260.

device types and host options

See: "FILENAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FILENAME ("*fileref*", "*filename*" <,*device-type*<,*host-options*<,*dir-ref*>>>)

Required Arguments

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.

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.

Optional Arguments

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 542 . 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 542 .

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 nonzero number if the operation was not successful.

Example

For an example of using the FILENAME function, see “Example: Obtaining Directory Information” on page 468.

FILEREF Function: Windows

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 Functions and CALL Routines: Reference*
“FILENAME Function” in *SAS Functions and CALL Routines: Reference*

Syntax

FILEREF("*fileref*")

Required Argument

fileref
specifies the fileref to be validated. Under Windows, *fileref* can also be a Windows environment variable. If the FILEREF function is used in a DATA step, *fileref* or the environment variable that you specify must be enclosed in quotation marks. If the FILEREF function is used in macro syntax, then *fileref* must not 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: Obtaining Directory Information" on page 468.

FINFO Function: Windows

Returns the value of an information item for an external file.

Category: External Files

Windows specifics: available *info-items*

See: "FINFO Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FINFO(*file-id*, *info-item*)

Required Arguments

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 item is a character value.

info-item for disk files can be one of these file information items:

- Create Time: *ddmmmyyyy:hh:mm:ss*

  Note: The Create Time date/time information item is localized to the site's locale. The date/time format might appear slightly different in the locale.

- Last Modified: *ddmmmyyyy:hh:mm:ss*
- Filename
- File size (bytes)
- 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: 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 6; " 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;
```

Example Code 17.2 *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 6; " 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 6; numopts=6
i=1 optname=Filename
optval=c:\tmp333
i=2 optname=RECFM
optval=V
i=3 optname=LRECL
optval=32767
i=4 optname=File Size (bytes)
optval=0
i=5 optname=Last Modified
optval=13Mar2007:13:12:23
i=6 optname=Create Time
optval=13Mar2007:13:12:23
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 Functions and CALL Routines: Reference*

FOPTNAME Function: Windows

Returns the name of an information item for an external file.

Category: External Files

Windows specifics: available information items

See: "FOPTNAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FOPTNAME(*file-id*, *nval*)

Required Arguments

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.

Table 17.1 *Information Items for Files*

nval	Single File	Pipe Files	Concatenated Files
1	Filename	Unnamed pipe access device	Filename
2	RECFM	PROCESS	RECFM
3	LRECL	RECFM	LRECL
4		LRECL	

Details

FOPTNAME returns a missing or null value if an invalid argument is used with FOPTNAME.

Example

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;
```

Example Code 17.3 *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=32767
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: Windows

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 Functions and CALL Routines: Reference*

Syntax

FOPTNUM(*file-id*)

Required Argument

file-id
specifies the identifier that was assigned when the file was opened, generally, by the FOPEN function.

Details

Six information items are available for files:

- Filename
- RECFM
- LRECL
- File Size (bytes)
- Last Modified
- Create Time

These information items are available for pipes:

- Unnamed pipe access device
- PROCESS
- RECFM
- LRECL

FOPTNUM returns the following values:

For files:
6

For pipes:
4

Example

For an example of the FOPTNUM functions, see "Example: Obtaining Directory Information" on page 468.

LIBNAME Function: Windows

Assigns or clears a libref for a SAS library.

Category: SAS File I/O

Windows specifics: behavior of the ' 'libref (a space between single quotation marks)

See: "LIBNAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

LIBNAME(*libref*<,*SAS-data-library*<,*engine*<,*options*>>>)

Required Argument

libref
specifies the libref that is assigned to a SAS library. Under Windows, the value of *libref* can be an environment variable.

Optional Arguments

SAS-data-library
specifies the physical name of the SAS 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 nonzero value, even if the function was successful. A nonzero value is returned if an error, warning, or note is produced. To determine whether 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: Windows

Causes SAS to wait for a piece of multimedia equipment to become active.

Category: Special

Windows specifics: all

Syntax

MCIPISLP(*number-of-seconds*)

Required Argument

number-of-seconds
specifies the number of seconds that 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 display enables 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: Windows" on page 486

MCIPISTR Function: Windows

Submits an MCI string command to a piece of multimedia equipment.

Category: Special

Windows specifics: all

Syntax

MCIPISTR(*MCI-string-command*)

Required Argument

MCI-string-command
is any valid SAS string; a character variable, a character literal enclosed in quotation marks, 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: Windows" on page 484

MODULE Function: Windows

Calls a specific routine or module that resides in an external dynamic link library (DLL).

Category: External Routines

Windows specifics: all

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.

Syntax

CALL MODULE(<*control string*>,*module*,*argument-1, ..., argument-n*);

num=**MODULEN**(<*control-string*>,*module*,*argument-1, ..., argument-n*);

char=**MODULEC**(<*control-string*> ,*module*,*argument-1, ..., argument-n*);

CALL MODULEI(<*control-string*> ,*moduleargument-1, ..., argument-n*);

num=**MODULEIN**(<*control-string*> ,*module*,*argument-1, ..., argument-n*)

char=**MODULEIC**(<*control-string*> ,*module*,*argument-1, ..., argument-n*);

Required Arguments

control-string

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 pass it as a constant.

argument
are the arguments to pass to the requested routine. Use the proper attributes for the arguments (numeric arguments for numeric attributes and character arguments for character attributes).

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. 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.

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.

CALL MODULEI, MODULEIN, and MODULEIC permit vector and matrix arguments; you can use them within the IML procedure.

See Also

"The SASCBTBL Attribute Table" on page 340

PEEKLONG Function: Windows

Stores the contents of a memory address in a numeric variable on 32-bit and 64-bit platforms.

Category: Special

Interaction: When a SAS server is in a locked-down state, the PEEKLONG function does not execute. For more information, see "LOCKDOWN" in *SAS Programmer's Guide: Essentials*.

Windows specifics: all

See: "PEEKLONG Function" in *SAS Functions and CALL Routines: Reference* and "PEEKCLONG Function" in *SAS Functions and CALL Routines: Reference*

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.

Syntax

PEEKLONG(*address*< ,*length*) >

Required Argument

address
specifies a character expression that is the memory address.

Optional Argument

length
specifies the length of the character data.

Details

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 use PEEKCLONG to access a character string.

RANK Function: Windows

Returns the position of a character in the ASCII collating sequence.

Category: Character

Windows specifics: Uses the ASCII sequence

See: "RANK Function" in *SAS Functions and CALL Routines: Reference*

Syntax

RANK(*x*)

Required Argument

x
is a character expression 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 expression.

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 characters are national characters and they vary from country to country.

SLEEP Function: Windows

Suspends execution of a SAS DATA step for a specified period of time.

Category: Special

Windows specifics: all

See: "SLEEP Function" in *SAS Functions and CALL Routines: Reference*

Syntax

SLEEP(*n*<,*unit*>)

Required Argument

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.

Optional Argument

unit

specifies the unit of seconds, as a multiple of 10, which is applied to *n*. For example, 1 corresponds to a second, and .001 to a millisecond. The default value 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 theSLEEP 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: Windows" on page 684

TRANSLATE Function: Windows

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 Functions and CALL Routines: Reference*

Syntax

TRANSLATE(*source*,*to-1*,*from-1* < ,...*to-n*,*from-n* >)

Syntax Description

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: Windows

Specifies the amount of time a SAS DATA step continues execution.

Category: Special

Windows specifics: This function runs only on Windows.
Uses the ASCII code sequence.

Syntax

WAKEUP(*until-when*)

Required Argument

until-when
specifies the time at which the WAKEUP function allow execution to continue.

Details

Use the WAKEUP function to specify the amount of time a DATA step continues 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, 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: Uses the WAKEUP Function with %SYSFUNC

This code wakes up SAS at 9:30 a.m.

```
%put %sysfunc(wakeup('09:30't));
```

Example 2: Delays Program Execution According to a Specified Date, Datetime, or Time Value

The code in this example tells SAS to delay execution of the program until 1:00 p.m. on January 1, 2022:

```
data _null_;
   slept=wakeup('01JAN2022:13:00:00'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 3: Delays 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 4: Uses 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: Windows" on page 684

18

SAS Informats under Windows

SAS Informats under Windows **497**
Overview of SAS Informats under Windows 497
Reading Binary Data **498**
Converting User-Written Informats from Earlier Releases to SAS 9.4 ..... **499**
Introduction to Converting User-Written Informats from Earlier Releases to SAS 9.4 499
Converting Version 6 User-Written Informats 499
Converting Version 5 User-Written Informats 499
Dictionary **500**
HEXw. Informat: Windows 500
$HEXw. Informat: Windows 501
IBw.d Informat: Windows 502
PDw.d Informat: Windows 504
PIBw.d Informat: Windows 505
RBw.d Informat: Windows 506
ZDw.d Informat: Windows 507

SAS Informats under Windows

Overview of 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 Formats and Informats: Reference*. 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 498 .

If you have informats that you created for use in earlier releases of SAS, see "Converting User-Written Informats from Earlier Releases to SAS 9.4" on page 499 for information about how to convert those informats for use with SAS 9.4.

Reading Binary Data

IBM mainframes, Hewlett Packard 9000, and most other UNIX systems store bytes in one order, called big-endian. Those that are based on Intel, 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 reads and chooses 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 enable 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 that 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 9.4

Introduction to Converting User-Written Informats from Earlier Releases to SAS 9.4

You must convert Release 6.04, Release 6.06, and Release 6.08 user-written informats and formats to their SAS 9.4 counterparts before you can use them in a SAS 9.4 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. [1]

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 9.4 to create the SAS 9.4 informats or formats. You must use the V604 engine in your SAS 9.4 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 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 9.4 counterparts before you can use them in a SAS 9.4 program. (This implies that

1. 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 9.4 if you no longer need to use them in previous releases.

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 9.4. You must use this procedure in Release 6 of SAS.

- Use the SAS Global Forum 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 9.4 to create the SAS 9.4 informats or formats.

Dictionary

HEXw. Informat: Windows

Converts hexadecimal positive binary values to fixed-point or floating-point binary values.

Category:	Numeric
Alignment:	Left
Windows specifics:	native floating-point representation
See:	"HEXw. Informat" in *SAS Formats and Informats: Reference*

Syntax

HEX*w*.

Required Argument

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.

Default 8

Range 1–16

Details

The HEX*w*. informat expects input that is not byte-reversed, not in Windows form. (The IB, PIB, and RB informats for binary numbers expect the bytes to be reversed.) You can use the HEX*w*. informat to read hexadecimal literals from SAS programs that were created in another environment.

See Also

- "$HEXw. Informat: Windows" on page 501
- "HEXw. Format: Windows" on page 449

$HEXw. Informat: Windows

Converts hexadecimal data to character data.

Category:	Character
Alignment:	Left
Windows specifics:	ASCII character-encoding system
See:	"$HEXw. Informat" in *SAS Formats and Informats: Reference*

Syntax

$HEX*w*.

Required Argument

w

specifies width of the input value.

Default 2

Range 1–132767

Details

The $HEX*w*. informat is like the HEX*w*. informat in that it reads values in which each hexadecimal character 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

- "HEXw. Informat: Windows" on page 500
- "$HEXw. Format: Windows" on page 450

IBw.d Informat: Windows

Reads integer binary (fixed-point) values.

Category: Numeric

Windows specifics: native floating-point representation

See: "IBw.d Informat" in *SAS Formats and Informats: Reference*

Syntax

IB*w*.*d*

Required Arguments

w

specifies the width of the input field.

Default 4

Range 1–8

d

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.

Range 0–10

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 developer website.

Comparisons

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. For 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

- "IBw.d Format: Windows" on page 451
- "Reading Binary Data" on page 498

PDw.d Informat: Windows

Reads packed decimal data.

Category: Numeric

Windows specifics: How values are interpreted as negative or positive

See: "PDw.d Informat" in *SAS Formats and Informats: Reference*

Syntax

PD*w*.*d*

Required Arguments

w

specifies the width of the input field.

Default 1

Range 1–16

d

specifies the power of 10 by which to divide the input value. If the data contain decimal points, then SAS ignores the *d* value.

Range 0–31

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

- “PDw.d Format: Windows” on page 453
- “Reading Binary Data” on page 498

PIBw.d Informat: Windows

Reads positive integer-binary (fixed-point) values.

Category: Numeric

Windows specifics: native byte-swapped integers

See: “PIBw.d Informat” in *SAS Formats and Informats: Reference*

Syntax

PIB*w.d*

Required Arguments

w

specifies the width of the input field.

Default 1

Range 1–8

d

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.

Range 0–10

Details

Positive integer binary values are the same as integer binary (see the informat “IBw.d Informat: Windows” on page 502), except that all values are treated as positive. Thus, the high-order bit is part of the value rather than the value’s sign.

Comparisons

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. For 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

- “PIBw.d Format: Windows” on page 455
- “Reading Binary Data” on page 498

RBw.d Informat: Windows

Reads real-binary (floating-point) data.

Category:	Numeric
Windows specifics:	native floating-point representation
See:	“RBw.d Informat” in *SAS Formats and Informats: Reference*

Syntax

RB*w.d*

Required Arguments

w

specifies the width of the input field.

Default 4

Range 2–8

d

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.

Range 0–10

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

- "RBw.d Format: Windows" on page 457
- "Reading Binary Data" on page 498

ZDw.d Informat: Windows

Reads zoned decimal data.

Category:	Numeric
Windows specifics:	Last byte includes the sign
See:	"ZDw.d Informat" in *SAS Formats and Informats: Reference*

Syntax

ZD*w.d*

Required Arguments

w

specifies the width of the input field.

Default 1

Range 1–32

d

specifies the power of 10 by which to divide the input value. If the data contain decimal points, then SAS ignores the *d* value.

Range 1–10

Details

This method 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:

Table 18.1 *Overprint Trailing Numeric Format*

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

"ZDw.d Format: Windows" on page 458

19

SAS Procedures under Windows

SAS Procedures under Windows **511**

Dictionary **511**
CATALOG Procedure Statement: Windows 511
CIMPORT Procedure Statement: Windows 512
CONTENTS Procedure Statement: Windows 514
CONVERT Procedure Statement: Windows 516
CPORT Procedure Statement: Windows 519
DATASETS Procedure Statement: Windows 520
OPTIONS Procedure Statement: Windows 524
PMENU Procedure Statement: Windows 525
PRINTTO Procedure Statement: Windows 527
SORT Procedure Statement: Windows 529

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*.

Dictionary

CATALOG Procedure Statement: Windows

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*;>

Required Argument

This syntax 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 the SASINITIALFOLDER location:

```
proc catalog catalog=sasuser.profile;
   contents file=myfile;
run;
```

CIMPORT Procedure Statement: Windows

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* <*options*> ;

Required Arguments

This is a simplified version of the CIMPORT procedure syntax. For the complete syntax and its explanation, see "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 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.

Note: You can use the MIGRATE procedure, beginning with SAS 9.1, to migrate a SAS library from a previous release.

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 the file created by PROC CPORT is not transferred in binary format, PROC CIMPORT cannot read the file. You receive the following message:

  ```
  ERROR: Given transport file is bad.
  ```

See Also

- "CPORT Procedure Statement: Windows" on page 519
- The MIGRATE procedure and cross-release compatibility at http://support.sas.com/migration/planning/files/regression.html and the *Base SAS Procedures Guide*.

CONTENTS Procedure Statement: Windows

Prints descriptions of the contents of one or more SAS library files.

Windows specifics: Engine Host Dependent Information output

See: "CONTENTS Procedure" in *Base SAS Procedures Guide*

Syntax

PROC CONTENTS <*options*> ;

Required Argument

options
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 set and prints the directory of the SAS library.

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 19.1 *Engine Host Dependent Information from PROC CONTENTS Using the V9 Engine*

SCHOOL DATASET

The CONTENTS Procedure

Data Set Name	WORK.SCHOOL	Observations	7
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	08/14/2014 09:06:22	Observation Length	40
Last Modified	08/14/2014 09:06:22	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	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1632
Obs in First Data Page	7
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\saskis\AppData\Local\Temp\SAS Temporary Files_TD4516_D73565_\school.sas7bdat
Release Created	9.0401M0
Host Created	W32_7PRO

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 256.

CONVERT Procedure Statement: Windows

Converts BMDP, OSIRIS system files, and SPSS export files to SAS data sets.

Windows specifics: All

Syntax

PROC CONVERT *product-specification* <*options*>

Required Arguments

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 that 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 option 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

Overview of the Convert Procedure

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 517. The BMDP, OSIRIS, and SPSS engines provide more extensive capabilities.

System files can be incompatible with the current version of PROC CONVERT, because the BMDP, OSIRIS and SPSS products are maintained by other companies or organizations. 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
Ensure that the translated names are unique. Variable names can sometimes be translated by SAS. To ensure that 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.

Comparisons

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.

CPORT Procedure Statement: Windows

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*<*options*> ;

Required Argument

This version is a simplified version of the CPORT procedure syntax. For the complete syntax and its explanation, see the CPORT procedure in "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 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.

Note: You can use the MIGRATE procedure, beginning with SAS 9.1, to migrate a SAS library from a previous release.

See Also

- "CIMPORT Procedure Statement: Windows" on page 512
- The MIGRATE procedure and cross-release compatibility at http://support.sas.com/rnd/migration/planning/files/regression.html

DATASETS Procedure Statement: Windows

Lists, copies, renames, and deletes SAS files and also manages indexes for and appends SAS data sets in a SAS library.

Windows specifics: Directory information; CONTENTS statement output

See: "DATASETS Procedure" in *Base SAS Procedures Guide*

Syntax

PROC DATASETS <*options(s)*> ;

CONTENTS <*options(s)*> ;

Required Argument

options

This version 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 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.

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.

Example:

The following SAS code creates two data sets, classes.grades and classes.majors, and executes PROC DATASETS using classes.majors as the input data set.

```
libname classes '.';
data classes.grades (label='First Data Set');
```

```
   input student year state $ grade1 grade2;
   label year='Year of Birth';
   format grade1 4.1;
   datalines;
1000 1980 NC 85 87
1042 1981 MD 92 92
1095 1979 PA 78 72
1187 1980 MA 87 94
;
data classes.majors(label='Second Data Set');
   input student $ year state $ grade1 grade2 major $;
   label state='Home State';
   format grade1 5.2;
   datalines;
1000 1980 NC 84 87 Math
1042 1981 MD 92 92 History
1095 1979 PA 79 73 Physics
1187 1980 MA 87 74 Dance
1204 1981 NC 82 96 French
;
proc datasets library=classes;
   contents data=majors directory;
run;
```

Output 19.2 *SAS Library Information from PROC DATASETS*

The SAS System

Directory	
Libref	CLASSES
Engine	V9
Physical Name	C:\Users\daharr
Filename	C:\Users\daharr

#	Name	Member Type	File Size	Last Modified
1	GRADES	DATA	128KB	08/20/2014 14:56:35
2	MAJORS	DATA	128KB	08/20/2014 14:56:35

The SAS System

The DATASETS Procedure

Directory	
Libref	CLASSES
Engine	V9
Physical Name	C:\Users\daharr
Filename	C:\Users\daharr

#	Name	Member Type	File Size	Last Modified
1	GRADES	DATA	128KB	08/20/2014 14:56:35
2	MAJORS	DATA	128KB	08/20/2014 14:56:35

The SAS System

The DATASETS Procedure

Data Set Name	CLASSES.MAJORS	Observations	5
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	08/20/2014 15:18:27	Observation Length	48
Last Modified	08/20/2014 15:18:27	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Second Data Set		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1361
Obs in First Data Page	5
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\daharr\majors.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
4	grade1	Num	8	5.2	
5	grade2	Num	8		
6	major	Char	8		
3	state	Char	8		Home State
1	student	Char	8		
2	year	Num	8		

See Also

"CONTENTS Procedure Statement: Windows" on page 514

OPTIONS Procedure Statement: Windows

Lists the current values of all SAS system options.

Windows specifics: Host options

See: "OPTIONS Procedure" in *SAS System Options: Reference*

Syntax

PROC OPTIONS <*options(s)*> ;

Required Argument

options
This version 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 can 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;
```

Example Code 19.1 *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 are examples. 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. However, 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 variable attributes in "SAS Options under Windows" on page 572 or Using SAS Software in Your Operating Environment in the SAS Help and Documentation.

See Also

"SAS Options under Windows" on page 572

PMENU Procedure Statement: Windows

Defines 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* <*options*> ;

ITEM '*menu-item*' <*options*> ;

ACCELERATE=*name-of-key*;

Required Argument

This version is a simplified version of the PMENU procedure syntax. For the complete syntax, see "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 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 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 use these alternate key combinations in your SAS program, the Edit menu shows the standard key combination. However, you can use either the standard or alternate key combination to activate the menu item.

PRINTTO Procedure Statement: Windows

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 <*options*> ;

Required Argument

options

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.

  Note: The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as filerefs.

- 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

This version is a simplified version of the PRINTTO procedure syntax. For the complete syntax and its explanation, see "PRINTTO Procedure" in *Base SAS Procedures Guide*

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 that 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 Statement: Windows

Sorts observations in a SAS data set by one or more variables, and 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

The Syncsort utility is no longer supported.

See: "SORT Procedure" in *Base SAS Procedures Guide*

Syntax

PROC SORT <*options*> <*collating-sequence-option*> ;

Required Arguments

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

Sort Procedure Syntax

This version is a simplified version of the SORT procedure syntax. For the complete syntax and its explanation, see the SORT procedure in "SORT Procedure" in *Base SAS Procedures Guide*

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, and your database. 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:

Syntax

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 1G 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.

TAGSORT Option

The TAGSORT option in the PROC SORT statement is useful in sorts when there might 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 can reduce temporary disk use, the processing time can be much higher. However, on PCs with limited available disk space, the TAGSORT option can 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): Reference 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** menu, select the `explorer` item, double-click the Sashelp library, and then double-click 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.

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 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): Reference Guide*.

- use the SAS sort.
- 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.

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 DATA Step Statements: Reference*. For more information about the NOBYSORTED system option, see the BYSORTED system option in *SAS System Options: Reference*.

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 19.3 *Output for 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): Reference Guide*
- "SORTSIZE System Option: Windows" on page 691
- "Improving Performance of the SORT Procedure" on page 254

20

SAS Statements under Windows

SAS Statements under Windows **535**

Dictionary **536**
ABORT Statement: Windows 536
ATTRIB Statement: Windows 537
FILE Statement: Windows 538
FILENAME Statement: Windows 542
FOOTNOTE Statement: Windows 549
%INCLUDE Statement: Windows 549
INFILE Statement: Windows 552
LENGTH Statement: Windows 555
LIBNAME Statement: Windows 556
SYSTASK Statement: Windows 560
TITLE Statement: Windows 563
WAITFOR Statement: Windows 564
X Statement: Windows 565

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 DATA Step Statements: Reference*.

Dictionary

ABORT Statement: Windows

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 DATA Step Statements: Reference*

Syntax

ABORT <ABEND | RETURN | CANCEL | NOLIST> <*n*>;

Optional Arguments

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
enables 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 values greater than 997 are used by SAS.

CANCEL <FILE>
causes the execution of the submitted statements to be canceled. Results depend on the method of operation.

NOLIST
suppresses the output of all variables to the SAS log.

n
is an integer value that enables you to specify a condition code:

- when used with the CANCEL argument, the value is placed in the SYSINFO automatic macro variable.

- when not used with the CANCEL argument, the error code that is returned by SAS is ERROR. The value of ERROR depends on the operating system. The condition code *n* is returned to the operating system as the final SAS system exit code.

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

- For more information, see "ABORT Statement" in *SAS DATA Step Statements: Reference*
- "Return Codes and Completion Status" on page 756

ATTRIB Statement: Windows

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 DATA Step Statements: Reference*

Syntax

ATTRIB *variable-list-1 attribute-list-1*...<*variable-list-n attribute-list-n*> ;

Syntax Description

Here is a simplified explanation of the ATTRIB statement syntax. For the complete syntax and its explanation, see the ATTRIB statement in "ATTRIB Statement" in *SAS DATA Step Statements: Reference*.

attribute-list

LENGTH=<**$**>***length***
specifies the length of the variables in *variable-list*. Under Windows, the length that you can specify for a numeric variable ranges from 3 to 8 bytes.

variable-list
names the variables that you want to associate with the attributes.

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: Windows

Specifies the current output file for PUT statements.

Valid in: a DATA step

Restrictions: The FILE statement cannot override or modify the device type that was set by a preceding FILENAME statement.

When SAS is in a locked-down state, the FILENAME statement is not available for files that are not in the lockdown path list. For more information, see "LOCKDOWN" in *SAS Programmer's Guide: Essentials*.

Windows specifics: Valid values for *file specification*; valid values for *encoding-value*; valid options for *host-option-list*

See: "FILE Statement" in *SAS DATA Step Statements: Reference*

Syntax

FILE *file-specification*<PERMISSION='*permission-value*'><ENCODING='*encoding-value*'>
<*option-list*> <*host-option-list*> ;

Required Argument

file-specification
can be any of the file specification forms discussed in "Referencing External Files" in the "Using External Files" section in *SAS Companion for Windows*.

The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as filenames.

Optional Arguments

PERMISSION='*permission-value*'
specifies permissions to set for the specified fileref.

'permission-value'

```
A::<trustee_type>::<permissions>
```

A
Access permissions. No other values are supported.

trustee_type

u user

g group (all groups)

o other (all others, including the user who generates the file.)

The permission values take the values r (Read), w (Write), and x (Execute), in that order. If you do not want to grant one of these permissions, enter a - in its place (for example, r-x or rw-).

To specify more than one set of permission values, separate them with a comma within the quotation marks.

To specify Read and Execute permission for all groups that you are a member of, specify a permission value of A::g::r-x. Specify a permission value of A::o::r-- to allow all users to have Read access to a file. The following code shows the permission value:

```
permission='A::o::r--'
```

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 on page 107 in *SAS National Language Support (NLS): Reference Guide*.

option-list
can be any of the options for the FILE statement that are valid in all operating environments.

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" and "Controlling Another Application Using DDE" in the "Using Dynamic Data Exchange" section in *SAS Companion for Windows*.

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" in the "Using Dynamic Data Exchange" section in *SAS Companion for Windows*.

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 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

Alternatively, you can specify a logical record length value by using the LRECL= system option.

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" in the "Using Dynamic Data Exchange" section in *SAS Companion for Windows*.

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. If LRECL is not specified, by default SAS uses the default value of 32767 bytes at a time from the file.
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 format 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. Here are the valid values:

CRLF
Carriage return line feed. Use TERMSTR=CRLF to write files that are formatted for Windows. CRLF is the default.

LF
Line feed. Use TERMSTR=LF to write files that are formatted for UNIX.

NL
New line. Use TERMSTR=NL to write files that are formatted for UNIX.

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

- “Named Pipe Examples” on page 331
- “DDE Examples” on page 746

FILENAME Statement: Windows

Associates a SAS fileref with an external file or an output device, disassociates a fileref and external file, or lists attributes of external files.

Valid in: anywhere in a SAS program

Restriction: When SAS is in a locked-down state, the following FILENAME statement and access methods are not available. Your server administrator can re-enable this access method so that it is accessible in the locked-down state.

- " EMAIL" on page 543
- " FTP" on page 543
- "Hadoop" on page 543
- HTTP
- " SOCKET" on page 544
- TCPIP
- "URL" on page 544

For more information, see "LOCKDOWN" in *SAS Programmer's Guide: Essentials*.

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 Global Statements: Reference*

Syntax

FILENAME *fileref* <*device-type*> '*external-file*' <PERMISSION='*permission-value*' >
<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*> ;

System Description

This version is a simplified version of the FILENAME statement syntax. For the complete syntax and its explanation, see the FILENAME statement in "FILENAME Statement" in *SAS Global Statements: Reference*.

fileref
is any valid fileref, as discussed in "Using a Fileref" in the "Using External Files" section in *SAS Companion for Windows*.

The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as filerefs.

For examples of using filerefs in member-name syntax (also called aggregate syntax), see "Assigning a Fileref to a Directory" on page 190 . For a discussion of the rules SAS uses when accessing files through filerefs, see "Understanding How Concatenated Directories Are Accessed" on page 196. See "SAS Names" in *SAS Programmer's Guide: Essentials* for length restrictions.

device-type

enables you to read and write data from devices rather than files. For information about the SFTP access method, see Configuring SSH Client Software in UNIX and Windows. The following values are valid:

CATALOG
reads a SAS catalog as an external flat file.

CLIPBOARD
reads text data from and writes text data to the clipboard on the host computer.

DATAURL
reads data from user-specified text by using the DATAURL access method.

DDE
reads data from and writes data to another application using Dynamic Data Exchange. For more information, see "DDE Syntax within SAS" on page 743.

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 Email Using SAS" on page 65.

Restriction When SAS is in a locked-down state, the FILENAME statement, EMAIL access method is not available. Your server administrator can re-enable this access method so that it is accessible in the locked-down state. For more information, see "LOCKDOWN" in *SAS Programmer's Guide: Essentials*.

FTP
lets you access information about other machines using TCP/IP. TCP/IP software and a WINSOCK.DLL must be 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 DATA Step Statements: Reference*.

Hadoop
accesses files on a Hadoop Distributed File System (HDFS) whose location is specified in a configuration file.

NAMEPIPE
writes data to a named pipe. For more information, see “Using Named Pipes ” on page 328.

PIPE
writes data to an unnamed pipe. For more information, see “Using Unnamed Pipes ” on page 326.

Requirement You must specify an *external-file* reference and the *external-file* reference must contain an absolute path and filename enclosed in quotation marks.

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 “SYSPRINT System Option: Windows” on page 703.

SFTP
accesses remote files by using the SFTP protocol.

SOCKET
lets you read and write information over a TCP/IP socket. TCP/IP software and a WINSOCK.DLL must be 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 DATA Step Statements: Reference* and *SAS Functions and CALL Routines: Reference*.

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 pathname is never shown to the user. If a physical pathname 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 204. The TAPE device-type keyword (documented in *SAS DATA Step Statements: Reference*) is not applicable to the Windows operating environment. If you use the TAPE device-type keyword in your SAS program under Windows, you receive an error message. The DISK device-type keyword is accepted under Windows. However, it is ignored because disk files are the default under Windows.

URL
accesses remote files by using the URL access method.

WebDAV
accesses remote files by using the WebDAV protocol.

ZIP
accesses ZIP files.

directory
specifies the directory that contains the files that you want to access.

external-file
can be any valid Windows file specification that is enclosed in quotation marks. For more information, see "Referencing External Files" on page 186.

PERMISSION='*permission-value*'
specifies permissions to set for the specified fileref.

'permission-value'

```
A::<trustee_type>::<permissions>
```

A
Access permissions. No other values are supported.

trustee_type

u user

g group (all groups)

o other (all others, including the user who generates the file.)

The permission values take the values r (Read), w (Write), and x (Execute), in that order. If you do not want to grant one of these permissions, enter a - in its place (for example, r-x or rw-).

To specify more than one set of permission values, separate them with a comma within the quotation marks.

To specify Read and Execute permission for all groups that you are a member of, specify a permission value of `A::g::r-x`. To allow all users to have Read access to a file, specify a permission value of `A::o::r--`. For example, specify that option in the code as follows:

```
permission='A::o::r--'
```

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 on page 107 in *SAS National Language Support (NLS): Reference 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.

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 744 and "Controlling Another Application Using DDE" on page 745.

COMTIMEOUT=*value*
controls how a communications port time-out is handled. A time-out 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 time-out occurs. This behavior is the default. The EOF character causes the current DATA step to terminate.

WAIT
instructs the communications port to wait forever for data. This value overrides the time-out. In this case, no record is returned to the DATA step until data are available. This action 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 time-out occurs. However, the DATA step does not terminate; it simply tries to read data again.

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.

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.

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 750.

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 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

Alternatively, you can specify a logical record length value by using the "LRECL= System Option" in *SAS System Options: Reference*

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 749.

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 either the PIPE device type or the NAMEPIPE device type. If LRECL is not specified, the default value of 32767 bytes is used to read from or write to the file.
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 format 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.

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. Here are the valid values:

CRLF
Carriage return line feed. Use TERMSTR=CRLF to write files that are formatted for Windows. CRLF is the default.

LF
Line feed. Use TERMSTR=LF to write files that are formatted for UNIX.

NL
New line. Use TERMSTR=NL to write files that are formatted for UNIX.

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.

Example: Referencing External Files

You can reference external files from a concatenated list of files or directories. The wildcard character * can be used in the FILENAME statement:

```
filename read ('c:\myfiles\*.*','c:\myotherfiles\abc.dat');
data new;
infile read;
input;
run;
```

See Also

"Advanced External I/O Techniques" on page 204

FOOTNOTE Statement: Windows

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 Global Statements: Reference*

Syntax

FOOTNOTE <*n*> <*'text'* | *"text"*>;

Optional Arguments

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: Windows

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 Global Statements: Reference*

Syntax

%INCLUDE *source* </<ENCODING='*encoding-value*'> <*host-options*> > ;

Required Argument

This version is a simplified version of the %INCLUDE statement syntax. For the complete syntax and its explanation, see the %INCLUDE statement in "%INCLUDE Statement" in *SAS Global Statements: Reference*.

source
describes the location of the information that you want to access. The two possible sources are a file specification or internal lines. The asterisk (*) cannot be used to specify keyboard entry if you use the Enhanced Editor in the Microsoft Windows operating environment. The file specification can be any of the file specification forms discussed in "Referencing External Files" on page 186.

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.

Optional Arguments

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 Chapter 3, "Using the SAS Editors under Windows," on page 107.

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 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

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, see "Using the NOTAB Option with DDE" on page 749.

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. If LRECL is not specified, by default SAS reads 32767 bytes at a time from the file.
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 format 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.

When using the keyboard method for preparing a program so that you can interrupt the current program's execution, the asterisk (*) cannot be used to specify keyboard entry if you use the Enhanced Editor in the Microsoft Windows operating environment. For more information about the keyboard entry method, see "%INCLUDE Statement" in *SAS Global Statements: Reference*.

INFILE Statement: Windows

Specifies an external file to read with an INPUT statement.

Valid in: a DATA step

Restriction: The INFILE statement cannot override or modify the device type that was set by a preceding FILENAME statement.

Windows specifics: Valid values for *encoding-value*, *file-specification*, and *host-options*

See: "INFILE Statement" in *SAS DATA Step Statements: Reference*

Syntax

INFILE *file-specification* <ENCODING='*encoding-value*'> <*options*><*host-options*>;

Required Argument

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 186. The reserved fileref CARDS enables the INFILE statement to reference instream data.

The words AUX, CON, NUL, PRN, LPT1 - LPT9, and COM1 - COM9 are reserved words under Windows. Do not use them as filerefs.

Optional Arguments

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 Chapter 3, "Using the SAS Editors under Windows," on page 107.

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.

COMTIMEOUT=*value*
controls how a communications port time-out is handled. A time-out 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 time-out occurs. This behavior is the default. The EOF character causes the current DATA step to terminate.

WAIT
instructs the communications port to wait forever for data. This value overrides the time-out. In this case, no record is returned to the DATA step until data are available. This action can cause your program to go into an infinite loop, so use this value with caution.

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.

ZERO
returns a record length of 0 bytes when a time-out occurs. However, the DATA step does not terminate; it simply tries to read data again.

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 750.

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 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

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 749.

PIPE
specifies an unnamed pipe.

Requirement You must specify a *file-specification* reference and the *file-specification* reference must contain an absolute path and filename enclosed in quotation marks.

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. If LRECL is not specified, by default SAS reads 256 bytes at a time from the file.
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 format 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. If termstr is not specified, a single LF or a CRLF function as the end of line character. If termstr=CRLF, then CRLF functions as the EOL character. The following values are valid under Windows:

CRLF
Carriage return line feed. Use TERMSTR=CRLF to read files that are formatted for Windows or DOS. CRLF is the default.

LF
Line feed. Use TERMSTR=LF to read files that are formatted for UNIX. If a file contains CRLF characters, the CR functions as part of the data and not an end of line character.

CR
Carriage Return. Use TERMSTR=CR if the end of line character is a CR.

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.

Example: Referencing External Files

You can reference external files from a concatenated list of files or directories. The wildcard character * can be used in the INFILE statement:

```
data new;
infile '("c:\myfiles\*.*","c:\myotherfiles\abc.dat")';
input;
run;
```

See Also

- “Named Pipe Examples” on page 331
- “DDE Examples” on page 746

LENGTH Statement: Windows

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 DATA Step Statements: Reference*

Syntax

LENGTH <*variable-1*> <...*variable-n*> <$> <*length*> <DEFAULT=*n*>;

Optional Arguments

variable
specifies one or more variables that are to be assigned a length. This includes any variables in the DATA step, including those dropped from the output data set.

$
specifies that the preceding variables are character variables.

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 can result in a loss of precision for the value of the variable.

See Also

"Length and Precision of Variables under Windows" on page 725

LIBNAME Statement: Windows

Associates a libref with a SAS library and lists file attributes for a SAS library.

Valid in:	anywhere in a SAS program
Windows specifics:	Valid values for *engine*; specifications for *SAS-data-library*
See:	"LIBNAME Statement" in *SAS Global Statements: Reference*

"LIBNAME Statement: V9 Engine" in *SAS V9 LIBNAME Engine: Reference*

Syntax

LIBNAME *libref* <*engine-name*> '('*'SAS-data-library-1'* <,...*'SAS-data-library-n'*>)' <MEMLIB>, <FILELOCKWAIT>;

LIBNAME *libref* _ALL_ LIST;

LIBNAME *libref* _ALL_ CLEAR;

Syntax Description

This version is a simplified version of the LIBNAME statement syntax. For the complete syntax and its explanation, see the LIBNAME statement in "LIBNAME Statement" in *SAS Global Statements: Reference*

libref
is any valid libref, as documented in *SAS DATA Step Statements: Reference*. See "SAS Names" in *SAS Programmer's Guide: Essentials* for length restrictions.

engine-name
is one of the following library engines supported under Windows:

V9	accesses SAS 9, SAS 9.1, SAS 9.2, SAS 9.3, and SAS 9.4 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. The V604 engine enables you to read from Release 6.03 and Release 6.04 SAS data sets directly from your 32-bit Windows SAS session. Release 6.03 and Release 6.04 SAS data sets are not compatible with the x64 64-bit environment and the Itanium 64-bit environment.
V604	accesses Release 6.03 and Release 6.04 data sets.
SAS/ACCESS engines (multiple)	access external data from other applications.
SPDE	accesses SPD Engine data sets.
CAS	accesses in-memory tables on the SAS Cloud Analytic Services (CAS) server.
CVP	expands the length of character variables to avoid truncation when a data set is transcoded.
JSON	accesses JSON data.

XMLV2 or XML	imports an XML document as a SAS data set or exports 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 export files.

For more information about these engines, see "Multi Engine Architecture" on page 157.

SAS-data-library
is the physical name of a SAS 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 library. When you specify multiple libraries, use parentheses around the first and last library pathnames. For more information about concatenated SAS libraries, see "Understanding How Multi-Folder SAS Libraries Are Accessed" on page 167.

MEMLIB
specifies to use extended server memory for this library. For more information about using extended memory, see "Memory-Based Libraries" on page 249.

FILELOCKWAIT=*n*
specifies the number of seconds SAS waits for a locked file to become available to another process. If the locked file is released before the number of seconds specified by *n*, then SAS locks the file for the current process and continues. If the file is still locked when the number of seconds has been reached, then SAS writes a "Locked File" error to the log and the DATA step fails.

Default 0

Range 0 – 600

Interactions Specifying the FILELOCKWAIT= option can have an adverse effect on one or more SAS/SHARE server and client sessions that are waiting for the release of a SAS file that is locked by another process. One or more wait conditions could lead to failed processes for a SAS/SHARE server and clients.

To prevent the possibility of a failed SAS/SHARE process, you can set FILELOCKWAIT=0, which cancels the amount of time that a SAS/SHARE server and clients would wait for the release of a locked file. Canceling the wait time would prevent a failed process. For more information, see "FILELOCKWAITMAX= System Option: Windows" on page 614.

Details

Overview of LIBNAME Statement

The LIBNAME statement associates a libref with a permanent SAS library. It can also be used to list the file attributes of a SAS library. The LIBNAME statement is also used to clear a libref. For more information, see "Clearing Librefs" on page 166.

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 library:

LIBNAME *libref* <*engine-name*> '*SAS-data-library*'

LIBNAME *libref* <*engine-name*> ('*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 libraries. The following LIBNAME statement results in the Data Library Attributes listing:

```
libname sashelp
list;
```

Output 20.1 *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\SASHome\SASFoundation\9.4\nls\en\SASCFG
      File Name= C:\Program Files\SASHome\SASFoundation\9.4\nls\en\SASCFG
        -Level 2-
      Engine=   V9
      Physical Name= C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp
      File Name= C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp
       . . .
      -Level 27-
      Engine=   V9
      Physical Name= C:\Program Files\SASHome\SASFoundation\9.4\webhound\sashelp
      File Name= C:\Program Files\SASHome\SASFoundation\9.4\webhound\sashelp
2    run;
```

See Also

- "LIBNAME Statement: SASEDOC" in *SAS Output Delivery System: User's Guide*
- "LIBNAME Statement" in *SAS Global Statements: Reference*
- "LIBNAME Statement: V9 Engine" in *SAS V9 LIBNAME Engine: Reference*

SYSTASK Statement: Windows

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-variable*>
<STATUS=*stat-variable*>
<SHELL<="*shell-command*"> > ;

SYSTASK LIST <_ALL_ | *taskname* > <STATE> <STATVAR>;

SYSTASK KILL *taskname* <*taskname*...>;

Syntax Description

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 quotation marks, repeat the quotation marks. For example:

```
systask command "find ""my text"" c:\mydir\myfile.sas"
```

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 automatically generates a name. If the task name contains a blank character, enclose the task name in quotation marks.

MNAME=*name-variable*
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 that you specified with TASKNAME into the variable that you specified with MNAME.

STATUS=*stat-variable*
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 quotation marks.

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 enables 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 cause your SAS application to end.

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 about 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 contains a nonzero value. For example, there might be insufficient resources to complete a task, or the SYSTASK statement can contain syntax errors. With the SYSTASK KILL statement, if one or more of the processes cannot be terminated, SYSRC is set to a nonzero 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.

Note: If you are using Microsoft Windows XP or later, the maximum length of the string that you can use at the command prompt is 8191 characters. For more information, see the article about restrictions on string length in command lines available at OS Limitation Workaround.

See Also

- "WAITFOR Statement: Windows" on page 564
- "X Statement: Windows" on page 565

TITLE Statement: Windows

prints up to ten title lines for SAS output.

Valid in: anywhere in a SAS program

Windows specifics: Maximum length of the title

See: "TITLE Statement" in *SAS Global Statements: Reference*

Syntax

TITLE <*n*> <'*text*' | "*text*">;

Optional Arguments

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 prints up to ten title lines on procedure output 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: Windows

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*>;

Syntax Description

taskname
specifies the name of the task(s) that you want to wait for. See "SYSTASK Statement: Windows" on page 560 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 finishes 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 suspends 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 contains a nonzero value. For example, the WAITFOR statement can 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 nonzero value if one of the following is true:

- 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: Windows" on page 560
- "X Statement: Windows" on page 565
- "XWAIT System Option: Windows" on page 722

X Statement: Windows

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 Global Statements: Reference*

Syntax

X <'*command*'> ;

Without Arguments

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.

Optional Argument

command

specifies a Windows command or a Windows application. This argument can be anything that 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 quotation marks, but this syntax is not required.

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: Windows" on page 722 and "XSYNC System Option: Windows" on page 721.

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.

Note: If you are using Microsoft Windows XP or later, the maximum length of the string that you can use at the command prompt is 8191 characters. For more information, see the article about restrictions on string length in command lines available at OS Limitation Workaround.

See Also

- "X Command: Windows" on page 444

- "XSYNC System Option: Windows" on page 721
- "XWAIT System Option: Windows" on page 722
- "CALL SYSTEM Routine: Windows" on page 464
- The %SYSEXEC statement in "Macro Statements" on page 733
- "Running Windows or MS-DOS Commands from within SAS" on page 45
- "Adding Applications to the Tools Menu" on page 92

21

SAS System Options under Windows

SAS Options under Windows ... **572**
Overview of SAS Options under Windows ... 572
Restricted Options ... 572

Displaying SAS Option Settings ... **573**

Changing SAS Option Settings ... **574**
Overview of Changing SAS Option Settings ... 574
Syntax for System Options in the SAS Invocation or SAS Configuration File ... 574
Syntax for Concatenating Libraries in SAS Options ... 574
Syntax for System Options in the OPTIONS Statement ... 574

Processing System Options That Are Set in Several Places ... **577**

SAS Options by Category ... **577**

Dictionary ... **586**
ACCESSIBILITY System Option: Windows ... 586
ALIGNSASIOFILES System Option: Windows ... 587
ALTLOG System Option: Windows ... 588
ALTPRINT System Option: Windows ... 589
APPEND System Option: Windows ... 590
AUTHPROVIDERDOMAIN System Option: Windows ... 592
AUTHSERVER System Option: Windows ... 592
AUTOEXEC System Option: Windows ... 593
AWSCONTROL System Option: Windows ... 594
AWSDEF System Option: Windows ... 596
AWSMENU System Option: Windows ... 597
AWSMENUMERGE System Option: Windows ... 597
AWSTITLE System Option: Windows ... 598
BUFNO System Option: Windows ... 599
BUFSIZE System Option: Windows ... 601
CATCACHE System Option: Windows ... 602
CLEANUP System Option: Windows ... 603
COMDEF System Option: Windows ... 604
CONFIG System Option: Windows ... 605

DEVICE System Option: Windows 606
ECHO System Option: Windows 607
EMAILDLG System Option: Windows 608
EMAILSYS System Option: Windows 609
ENGINE System Option: Windows 610
ENHANCEDEDITOR System Option: Windows 612
FILELOCKWAIT System Option: Windows 613
FILELOCKWAITMAX= System Option: Windows 614
FILTERLIST System Option: Windows 615
FONT System Option: Windows 616
FONTALIAS System Option: Windows 618
FONTSLOC System Option: Windows 619
FORMCHAR System Option: Windows 620
FULLSTIMER System Option: Windows 621
HELPHOST System Option: Windows 623
HELPINDEX System Option: Windows 624
HELPLOC System Option: Windows 625
HELPREGISTER System Option: Windows 627
HELPTOC System Option: Windows 628
HOSTINFOLONG System Option: Windows 630
HOSTPRINT System Option: Windows 631
ICON System Option: Windows 632
INITSTMT System Option: Windows 633
INSERT System Option: Windows 633
JREOPTIONS System Option: Windows 635
LINESIZE System Option: Windows 636
LOADMEMSIZE System Option: Windows 637
LOG System Option: Windows 638
MAPS System Option: Windows 640
MAXMEMQUERY System Option: Windows 641
MEMBLKSZ System Option: Windows 642
MEMCACHE System Option: Windows 643
MEMLIB System Option: Windows 644
MEMMAXSZ System Option: Windows 645
MEMSIZE System Option: Windows 647
MSG System Option: Windows 648
MSGCASE System Option: Windows 649
MSYMTABMAX System Option: Windows 650
MVARSIZE System Option: Windows 651
NEWS System Option: Windows 652
NUMKEYS System Option: Windows 653
NUMMOUSEKEYS System Option: Windows 654
OBS System Option: Windows 655
OPLIST System Option: Windows 656
PAGENO System Option: Windows 656
PAGESIZE System Option: Windows 657
PAPERTYPE System Option: Windows 659
PATH System Option: Windows 660
PFKEY System Option: Windows 660
PRIMARYPROVIDERDOMAIN= System Option: Windows 664
PRINT System Option: Windows 664
PRNGETLIST System Option: Windows 665
PRTABORTDLGS System Option: Windows 666
PRTPERSISTDEFAULT System Option: Windows 667

PRTSETFORMS System Option: Windows ... 668
REALMEMSIZE System Option: Windows ... 669
REGISTER System Option: Windows ... 670
RESOURCESLOC System Option: Windows ... 671
RSASUSER System Option: Windows ... 672
RTRACE System Option: Windows ... 673
RTRACELOC System Option: Windows ... 674
SASAUTOS System Option: Windows ... 675
SASCONTROL System Option: Windows ... 676
SASHELP System Option: Windows ... 677
SASINITIALFOLDER System Option: Windows ... 679
SASUSER System Option: Windows ... 680
SCROLLBARFLASH System Option: Windows ... 681
SET System Option: Windows ... 682
SGIO System Option: Windows ... 683
SLEEPWINDOW System Option: Windows ... 684
SORTCUT System Option: Windows ... 685
SORTCUTP System Option: Windows ... 687
SORTDEV System Option: Windows ... 688
SORTNAME System Option: Windows ... 689
SORTPARM System Option: Windows ... 690
SORTPGM System Option: Windows ... 691
SORTSIZE System Option: Windows ... 691
SPLASH System Option: Windows ... 693
SPLASHLOC System Option: Windows ... 694
STIMEFMT System Option: Windows ... 695
STIMER System Option: Windows ... 699
SYSGUIFONT System Option: Windows ... 700
SYSIN System Option: Windows ... 701
SYSPARM System Option: Windows ... 702
SYSPRINT System Option: Windows ... 703
SYSPRINTFONT System Option: Windows ... 705
TOOLDEF System Option: Windows ... 707
TOOLSMENU System Option ... 708
UNIVERSALPRINT System Option ... 709
UPRINTMENUSWITCH System Option: Windows ... 710
USER System Option: Windows ... 711
USERICON System Option: Windows ... 712
VERBOSE System Option: Windows ... 713
VIEWMENU System Option ... 714
WEBUI System Option: Windows ... 715
WINDOWSMENU System Option: Windows ... 716
WORK System Option: Windows ... 717
XCMD System Option: Windows ... 719
XMIN System Option: Windows ... 720
XSYNC System Option: Windows ... 721
XWAIT System Option: Windows ... 722

SAS Options under Windows

Overview of SAS 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, using PROC OPTLOAD or the DMOPTLOAD command (PROC OPTLOAD and the DMOPTLOAD command load options settings that were previously saved in a SAS data set), in an OPTIONS statement (either in a SAS program or in a SAS autoexec file), in the System Options window, or in SCL programs.

If you save data in the configuration file, the default location is `C:\Program Files\SASHome2\SASFoundation\9.4\nls\language\sasv9.cfg` where language is the language of your locale (for example, en for English).

When 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).

Restricted Options

Restricted options are system options whose values are determined by the site administrator and cannot be overridden. The site administrator can create a restricted options table that specifies the option values that are restricted when SAS starts. Any attempt to modify a system option that is listed in the restricted options table results in a message to the SAS log indicating that the system option has been restricted by the site administrator and cannot be updated. For more information about restricted options, see “Restricted Options” in *SAS System Options: Reference*.

Global restricted option specifications should be located in a file in the **!SASROOT\rstropts\rsasv9.cfg** path. User restricted option specifications should be located in a file in the **!SASROOT\rstropts\users\userid_rsasv9.cfg** path, where user-ID is the user's system ID (that is, jomaxw).

- COMDEF
- FILELOCKWAITMAX
- MEMCACHE
- MEMLIB
- PATH
- RESOURCESLOC
- SASCONTROL
- SGIO
- TOOLDEF
- XCMD

Displaying SAS 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 System Options window displays the settings of many SAS system options, including the invocation and configuration options. You cannot edit these options from the Options window. To open the System Options window, enter **Tools** ⇨ **Options** ⇨ **System**, or type **options** in the command area and submit the command.

Changing SAS Option Settings

Overview of Changing SAS 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
- using PROC OPTLOAD or the DMOPTLOAD command to load a previous set of option values saved to a SAS data set.
- as part of the OPTIONS statement from within your SAS session
- using the interactive System Options window
- 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). 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 about the OPTIONS statement, see *SAS DATA Step Statements: Reference*.

You can set the BUFNO and MEMSIZE options in the configuration file. MEMSIZE cannot be set after SAS starts.

```
-BUFNO 10
-MEMSIZE 4G
```

You can also set the BUFNO option using the OPTION statement:

```
OPTIONS BUFNO=10;
```

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 quotation marks when you use them in the SAS command or in the SAS configuration file. The quotation marks are especially necessary when the file or pathname that you are specifying contains a space or single quotation mark 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. The file can contain SAS comments. For example, a configuration file named MySASConfig.CFG can contain these option specifications (among others):

```
-nocenter
-noxwait
-pagesize 60
```

All SAS system options can appear in a SAS configuration file. For more information about SAS configuration files, see “SAS Configuration Files ” on page 28 .

Syntax for Concatenating Libraries in SAS 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 quotation marks, 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 or load a previously saved set of option values using PROC OPTLOAD or the DMOPTLOAD command. For more information about the OPTIONS statement, see *SAS DATA Step Statements: Reference*.

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 quotation marks 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 quotation marks 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, use PROC OPTIONS

```
option=optname define;
run;
```

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 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.

SAS Options by Category

The categories for SAS system options correspond to the SAS system option groups and subgroups:

Table 21.1 *Category and Definitions*

Category	Definition
Communications: E-mail	options associated with sending and receiving email using SAS
Communications: Networking and Encryption	options related to remote communication, shared settings, and encryption
Communications: Metadata	options to configure SAS to use metadata
Environment Control: Display	options to set SAS windows and display preferences
Environment Control: Error Handling	options associated with error conditions and error messages

Category	Definition
Environment Control: Files	options to set SAS library and file location preferences
Environment Control: Help	options used to configure the SAS Help
Environment Control: Initialization and Operation	options that establish the SAS operating environment
Environment Control: Language Control	options to set language and translation preferences
Files: External Files	options that define how to process files that are not created by SAS
Files: SAS Files	options that define how to process SAS files
Input Control: Data Processing	options for data entry and data processing preferences
Input Control: Data Quality	options to configure the SAS® Data Quality Server
Graphics: Driver Settings	options that define devices, graphics, and map preferences
Log and Procedure Output Control: SAS Log	options that control the display of messages that are written to the SAS log
Log and Procedure Output Control: Procedure Output	options that define procedure output and display preferences
Log and Procedure Output Control: SAS Log and Procedure Output	options that control both SAS log and procedure output preferences
Log and Procedure Output Control: ODS Printing	options to define preferences for printing to ODS destinations
Log and Procedure Output Control: PDF	options to define preferences for PDF files
Log and Procedure Output Control: SVG	options to define preferences for SVG files
Log and Procedure Output Control: Animation	options to define preferences for Animating SVG files

Category	Definition
Macro: SAS Macro	options that define SAS macro preferences
Sort: Procedure Options	options that define preferences for sorting SAS files
System Administration: Installation	options for defining site installation settings
System Administration: Memory	options that define computer memory preferences
System Administration: Performance	options that define performance preferences
System Administration: Code Generation	options that define preferences for generating SAS language statements
System Administration: Security	options for defining security settings
System Administration: SQL	options that define settings for the SQL procedure

Category	Language Elements	Description
Communications: E-mail	EMAILDLG System Option: Windows (p. 608)	Specifies whether to use the native email dialog box provided by your email application or the email dialog box provided by SAS.
	EMAILSYS System Option: Windows (p. 609)	Specifies the email protocol to use for sending electronic mail.
Environment Control: Display	AWSCONTROL System Option: Windows (p. 594)	Specifies whether the main SAS window includes a title bar, a system control menu, and minimize maximize buttons.
	AWSDEF System Option: Windows (p. 596)	Specifies the location and dimensions of the main SAS window when SAS initializes.
	AWSMENU System Option: Windows (p. 597)	Specifies whether to display the menu bar in the main SAS window.
	AWSMENUMERGE System Option: Windows (p. 597)	Specifies whether to embed menu items that are specific to Windows in the main menus.
	AWSTITLE System Option: Windows (p. 598)	Replaces the default text in the main SAS title bar.
	COMDEF System Option: Windows (p. 604)	Specifies the location where the SAS Command window is displayed.

Category	Language Elements	Description
	ENHANCEDEDITOR System Option: Windows (p. 612)	Specifies whether to enable the Enhanced Editor during SAS invocation.
	FILTERLIST System Option: Windows (p. 615)	Specifies an alternative set of file filter specifications to use for the Open and Save As dialog boxes.
	FONT System Option: Windows (p. 616)	Specifies a font to use for SAS windows.
	FONTSLOC System Option: Windows (p. 619)	Specifies the location of the SAS fonts that are loaded during the SAS session.
	ICON System Option: Windows (p. 632)	Minimizes the SAS window.
	PRTABORTDLGS System Option: Windows (p. 666)	Specifies when to display the Print Abort dialog box.
	PRTSETFORMS System Option: Windows (p. 668)	Specifies whether to include the Use Forms check box in the Print Setup dialog box.
	REGISTER System Option: Windows (p. 670)	Adds an application to the Tools menu in the main SAS window.
	RESOURCESLOC System Option: Windows (p. 671)	Specifies a directory location of the files that contain SAS resources.
	SASCONTROL System Option: Windows (p. 676)	Specifies whether the SAS application windows include system and control menus and minimize and maximize buttons.
	SCROLLBARFLASH System Option: Windows (p. 681)	Specifies whether to allow the mouse or keyboard to focus on a scroll bar.
	SLEEPWINDOW System Option: Windows (p. 684)	Enables or disables the SLEEP window.
	SPLASH System Option: Windows (p. 693)	Specifies whether to display the splash screen (logo screen) when SAS starts.
	SPLASHLOC System Option: Windows (p. 694)	Specifies the location of the splash screen bitmap that appears when SAS starts.
	SYSGUIFONT System Option: Windows (p. 700)	Specifies a font to use for the button text and the descriptive text.
	TOOLDEF System Option: Windows (p. 707)	Specifies the Toolbox display location.
	TOOLSMENU System Option (p. 708)	Specifies whether the Tools menu is included in SAS windows.
	USERICON System Option: Windows (p. 712)	Specifies the pathname of the resource file associated with your user-defined icon.

Category	Language Elements	Description
	VIEWMENU System Option (p. 714)	Specifies whether the View menu is included in SAS windows.
	WINDOWSMENU System Option: Windows (p. 716)	Specifies to include or suppress the Window menu in windows that display menus.
	XCMD System Option: Windows (p. 719)	Specifies that the X command is valid in the current SAS session.
	XMIN System Option: Windows (p. 720)	Specifies to open the application specified in the X command in a minimized state or in the default active state.
	XSYNC System Option: Windows (p. 721)	Controls whether an X command or statement executes synchronously or asynchronously.
	XWAIT System Option: Windows (p. 722)	Specifies whether you have to type EXIT at the DOS prompt before the DOS shell closes.
Environment Control: Error Handling	CLEANUP System Option: Windows (p. 603)	Specifies how to handle an out-of-resource condition.
Environment Control: Files	ALTLOG System Option: Windows (p. 588)	Specifies a destination for a copy of the SAS log.
	ALTPRINT System Option: Windows (p. 589)	Specifies the destination for the copies of the output files from SAS procedures.
	AUTOEXEC System Option: Windows (p. 593)	Specifies the SAS autoexec file.
	LOG System Option: Windows (p. 638)	Specifies a destination for a copy of the SAS log when SAS is running in batch mode.
	MSG System Option: Windows (p. 648)	Specifies the library that contains the SAS error messages.
	NEWS System Option: Windows (p. 652)	Specifies a file that contains messages to be written to the SAS log.
	PRINT System Option: Windows (p. 664)	Specifies a destination for SAS output when running in batch mode.
	RSASUSER System Option: Windows (p. 672)	Controls whether members of the Sasuser data library can be opened for update or for Read-Only access.
	RTRACELOC System Option: Windows (p. 674)	Specifies the pathname of the file to which the list of resources that are read or loaded during a SAS session is written.
	SASAUTOS System Option: Windows (p. 675)	Specifies the autocall macro library.

Category	Language Elements	Description
	SASHELP System Option: Windows (p. 677)	Specifies the directory or directories to be searched for SAS default forms, device lists, dictionaries, and other entries in the Sashelp catalog.
	SASINITIALFOLDER System Option: Windows (p. 679)	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.
	SASUSER System Option: Windows (p. 680)	Specifies the name of the Sasuser library.
	SET System Option: Windows (p. 682)	Defines a SAS environment variable.
	SYSIN System Option: Windows (p. 701)	Specifies a batch mode source file.
	SYSPARM System Option: Windows (p. 702)	Specifies a character string that can be passed to SAS programs.
	USER System Option: Windows (p. 711)	Specifies the name of the default permanent SAS library.
	WORK System Option: Windows (p. 717)	Specifies the location of the Work library.
Environment Control: Help	HELPINDEX System Option: Windows (p. 624)	Specifies one or more index files for the SAS Help and Documentation.
	HELPLOC System Option: Windows (p. 625)	Specifies the location of Help files that are used to view SAS Help and Documentation using Microsoft HTML Help.
	HELPREGISTER System Option: Windows (p. 627)	Registers help files to access from the main SAS window Help menu.
	HELPTOC System Option: Windows (p. 628)	Specifies the table of contents files for the SAS Help and Documentation.
Environment Control: Initialization and Operation	AUTHPROVIDERDOMAIN System Option: Windows (p. 592)	Associates a domain suffix with an authentication provider.
	AUTHSERVER System Option: Windows (p. 592)	Specifies the authentication domain server to search for secure server logins.
	INITSTMT System Option: Windows (p. 633)	Specifies a SAS statement to be executed after any statements in the autoexec file and before any statements from the SYSIN= file.
	JREOPTIONS System Option: Windows (p. 635)	Identifies Java Runtime Environment (JRE) options for SAS.
	PRIMARYPROVIDERDOMAIN= System Option: Windows (p. 664)	Specifies the domain name of the primary authentication provider.

Category	Language Elements	Description
Environment Control: SAS Macro	SASAUTOS System Option: Windows (p. 675)	Specifies the autocall macro library.
Files: SAS Files	BUFNO System Option: Windows (p. 599)	Specifies the number of buffers to be allocated for processing SAS data sets.
	BUFSIZE System Option: Windows (p. 601)	Specifies the permanent buffer page size for output SAS data sets.
	CATCACHE System Option: Windows (p. 602)	Specifies the number of SAS catalogs to keep open.
	ENGINE System Option: Windows (p. 610)	Specifies the default access method to use for SAS libraries.
	FILELOCKWAITMAX= System Option: Windows (p. 614)	Sets an upper limit on the amount of time that SAS waits for a locked file.
	MEMCACHE System Option: Windows (p. 643)	Specifies to use the memory-based libraries as a SAS file cache.
	MEMLIB System Option: Windows (p. 644)	Specifies to process the Work library as a memory-based library.
	OBS System Option: Windows (p. 655)	Specifies when to stop processing observations or records.
	SGIO System Option: Windows (p. 683)	Activates the Scatter/Gather I/O feature.
Graphics: Driver Settings	DEVICE System Option: Windows (p. 606)	Specifies a device driver for graphics output for SAS/GRAPH software.
	FONTALIAS System Option: Windows (p. 618)	Assigns a Windows font to one of the SAS fonts.
	MAPS System Option: Windows (p. 640)	Specifies the name of the SAS library that holds the SAS/GRAPH map data sets.
Input Control: Data Processing	ACCESSIBILITY System Option: Windows (p. 586)	Enables the accessibility features on the Customize Tools dialog box.
	ALIGNSASIOFILES System Option: Windows (p. 587)	Aligns the pages of data in a SAS data set.
	NUMKEYS System Option: Windows (p. 653)	Controls the number of available function keys.
	NUMMOUSEKEYS System Option: Windows (p. 654)	Specifies the number of mouse buttons SAS is displayed in the KEYS window.

Category	Language Elements	Description
	PFKEY System Option: Windows (p. 660)	Specifies which set of function keys to designate as the primary set of function keys.
	WEBUI System Option: Windows (p. 715)	Specifies to enable web enhancements.
Log and Procedure Output Control: ODS Printing	PAPERTYPE System Option: Windows (p. 659)	Specifies to a printer the type of paper to use for printing.
	PRNGETLIST System Option: Windows (p. 665)	Specifies whether printers that are attached to the system are recognized.
	PRTPERSISTDEFAULT System Option: Windows (p. 667)	Specifies to use the same destination printer from SAS session to SAS session.
	UNIVERSALPRINT System Option (p. 709)	Specifies whether to enable menus for Universal Printing and to set up printing defaults.
	UPRINTMENUSWITCH System Option: Windows (p. 710)	Enables the universal print commands on the File menu.
Log and Procedure Output Control: Procedure Output	FORMCHAR System Option: Windows (p. 620)	Specifies the default output formatting characters.
	HOSTPRINT System Option: Windows (p. 631)	Specifies that the Windows Print Manager is to be used for printing.
	PAGENO System Option: Windows (p. 656)	Resets the page number.
	PRNGETLIST System Option: Windows (p. 665)	Specifies whether printers that are attached to the system are recognized.
	SYSPRINT System Option: Windows (p. 703)	Specifies a destination printer for printing SAS output.
	SYSPRINTFONT System Option: Windows (p. 705)	Specifies the font to use when SAS is printing to the current default printer.
Log and Procedure Output Control: SAS Log	ECHO System Option: Windows (p. 607)	Specifies a message to be echoed to the SAS log while initializing SAS.
	FULLSTIMER System Option: Windows (p. 621)	Specifies whether to write all available system performance statistics to the SAS log.
	HOSTINFOLONG System Option: Windows (p. 630)	Specifies to write additional operating-environment information in the SAS log when SAS starts.
	MSGCASE System Option: Windows (p. 649)	Specifies whether notes, warnings, and error messages that are generated by SAS are displayed in uppercase characters.

Category	Language Elements	Description
	OPLIST System Option: Windows (p. 656)	Specifies whether the settings of the SAS system options are written to the SAS log.
	RTRACE System Option: Windows (p. 673)	Produces a list of resources that are read or loaded during a SAS session.
	STIMEFMT System Option: Windows (p. 695)	Specifies the format that is used to display the time on FULLSTIMER and STIMER output.
	STIMER System Option: Windows (p. 699)	Writes a subset of system performance statistics to the SAS log.
	VERBOSE System Option: Windows (p. 713)	Controls whether SAS writes the settings of all the system options specified in the configuration file to either the terminal or the batch log.
Log and Procedure Output Control: SAS Log and Procedure Output	LINESIZE System Option: Windows (p. 636)	Specifies the line size of SAS Log and Output windows.
	PAGESIZE System Option: Windows (p. 657)	Specifies the number of lines that compose a page of SAS output.
Macro: SAS Macro	MSYMTABMAX System Option: Windows (p. 650)	Specifies the maximum amount of memory available to the macro variable symbol table(s).
	MVARSIZE System Option: Windows (p. 651)	Specifies the maximum size for in-memory macro variables.
SAS Files	FILELOCKWAIT System Option: Windows (p. 613)	Sets the number of seconds that SAS waits for a locked file to become available.
Sort: Procedure Options	SORTCUT System Option: Windows (p. 685)	Specifies the data size in number of observations above which SAS uses the host sort instead of the internal SAS sort.
	SORTCUTP System Option: Windows (p. 687)	Specifies the data size in bytes above which SAS uses the host sort instead of the internal SAS sort.
	SORTDEV System Option: Windows (p. 688)	Specifies the pathname to temporary files that are created by the host sort utility.
	SORTPARM System Option: Windows (p. 690)	Specifies the parameters for the host sort utility.
	SORTPGM System Option: Windows (p. 691)	Specifies whether to use the SAS sort utility or to let SAS choose the sort utility.
	SORTSIZE System Option: Windows (p. 691)	Specifies the amount of memory that is available to the SORT procedure.
System Administration: Installation	CONFIG System Option: Windows (p. 605)	Specifies the configuration file that is used when initializing or overriding the values of SAS system options.

Category	Language Elements	Description
	PATH System Option: Windows (p. 660)	Specifies one or more search paths for SAS executable files.
System Administration: Memory	LOADMEMSIZE System Option: Windows (p. 637)	Specifies a suggested amount of memory needed for executable programs loaded by SAS.
	MAXMEMQUERY System Option: Windows (p. 641)	Specifies the limit on the maximum amount of memory that is allocated for procedures.
	MEMBLKSZ System Option: Windows (p. 642)	Specifies the memory block size for memory-based libraries for Windows operating environments.
	MEMMAXSZ System Option: Windows (p. 645)	Specifies the maximum amount of memory to allocate for using memory-based libraries in Windows operating environments.
	MEMSIZE System Option: Windows (p. 647)	Specifies the limit on the amount of virtual memory that can be used during a SAS session.
	REALMEMSIZE System Option: Windows (p. 669)	Specifies the amount of real memory SAS can expect to allocate.
	SORTSIZE System Option: Windows (p. 691)	Specifies the amount of memory that is available to the SORT procedure.

Dictionary

ACCESSIBILITY System Option: Windows

Enables the accessibility features on the Customize Tools dialog box.

Valid in: configuration file, SAS invocation

Category: Input Control: Data Processing

PROC OPTIONS GROUP= INPUTCONTROL

Default: STANDARD

Windows specifics: all

Syntax

-ACCESSIBILITY STANDARD | EXTENDED

Required Arguments

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 **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 **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.

ALIGNSASIOFILES System Option: Windows

Aligns the pages of data in a SAS data set.

Valid in:	configuration file, SAS invocation
Category:	Input Control: Data Processing
PROC OPTIONS GROUP=	SASFILES
Default:	OFF
Windows specifics:	ALL

Syntax

-ALIGNSASIOFILES

-NOALIGNSASIOFILES

Without Arguments

There are no required arguments.

Details

A SAS data set consists of a header followed by one or more pages of data. Normally, the header is 1K on a Windows box and 8K on UNIX. ALIGNSASIOFILES forces the header to be the same size as the data page, allowing the pages to align to boundaries that allow for more efficient I/O.

ALTLOG System Option: Windows

Specifies a destination for a copy of the SAS log.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES, LOGCONTROL
Default:	NOALTLOG
Windows specifics:	*destination* must resolve to a valid Windows path or file name

Syntax

-ALTLOG *destination*

-NOALTLOG

Required Arguments

ALTLOG *destination*
specifies a destination for a copy of the SAS log. The *destination* argument can be a valid Windows pathname or file name (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
specifies that the SAS log is not copied.

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.

Using directives in the value of the ALTLOG system option enables you to control when logs are open and closed and how they are named, based on real-time events, such as time, month, day of week. For a list of directives see the LOGPARM= system option in the *SAS System Options: Reference*.

When SAS is started with the OBJECTSERVER and NOTERMINAL system options and no log is specified, SAS discards all log and alternate log messages.

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 228
- The SAS Log

ALTPRINT System Option: Windows

Specifies the destination for the copies of the output files from SAS procedures.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	NOALTPRINT
Windows specifics:	*destination* must resolve to a valid Windows pathname or file name

Syntax

-ALTPRINT *file-specification*

-NOALTPRINT

Required Arguments

ALTPRINT *file-specification*
specifies the destination for a copy of the SAS procedure output file. The *file-specification* argument can be a valid Windows pathname or file name (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 file name is SAS.

NOALTPRINT
does not create a copy of the SAS procedure output file.

Details

The ALTPRINT system option specifies the destination for the copies of output files from SAS procedures. 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 228
- "Printing" on page 208

APPEND System Option: Windows

Used when SAS starts, appends the specified value at the end of the specified system option.

Valid in:	configuration file, SAS invocation, OPTIONS statement, System Options window
PROC OPTIONS GROUP=	ENVFILES

Default: none

Windows specifics: SAS invocation syntax

Syntax

-APPEND *system-option argument*

Required Arguments

system-option
can be AUTOEXEC, CMPLIB, FMTSEARCH, HELPLOC, MAPS, MSG, SAMPLOC, SASAUTOS, SASHELP, SASSCRIPT, or SET.

argument
specifies a new pathname or an environment setting that you want to append to the current value of *system-option*. The following example shows that a library is being appended to the FMTSEARCH option:

```
-set APFMTLIB "SASEnvironment/SASFormats"
-append fmtsearch APFMTLIB
```

Details

By default, if you specify the AUTOEXEC, CMPLIB, FMTSEARCH, HELPLOC, MAPS, MSG, SASAUTOS, SASSCRIPT, SASHELP, or SET system option more than one time, the last value that is specified is the value that SAS uses. If you want to append pathnames to the pathnames that were already specified by one of these options, you must use the APPEND or INSERT system option to append the new pathname. For example, if you entered the following SAS command, the only location that SAS looks for help files is **`c:\app2\help`**, and the output of PROC OPTIONS shows only **`C:\app2\help`**:

```
sas -helploc "c:\app1\help" -helploc "c:\app2\help"
```

If you want SAS to look in both locations for help files, you can append the new location to the value of the HELPLOC option using the APPEND option:

```
sas -helploc "c:\app1\help" -append helploc "c:\app2\help"
```

For the value of the HELPLOC option, PROC OPTIONS now shows

```
("c:\app1\help" "c:\app2\help")
```

See Also

- "Changing an Option Value By Using the INSERT and APPEND System Options" in *SAS System Options: Reference*
- "APPEND= System Option" in *SAS System Options: Reference*

- "INSERT= System Option" in *SAS System Options: Reference*

AUTHPROVIDERDOMAIN System Option: Windows

Associates a domain suffix with an authentication provider.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Initialization and Operation
PROC OPTIONS GROUP=	EXECMODES
Alias:	AUTHPD
See:	"AUTHPROVIDERDOMAIN System Option" in *SAS System Options: Reference*

Syntax

AUTHPROVIDERDOMAIN <provider: domain>

AUTHPROVIDERDOMAIN <(provider-1: domain-1<, ...provider: domain-n>)>

AUTHSERVER System Option: Windows

Specifies the authentication domain server to search for secure server logins.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Initialization and Operation
PROC OPTIONS GROUP=	EXECMODES
Default:	local and trusted servers
Windows specifics:	all

Syntax

-AUTHSERVER <" " | '*domain-name*' | '.'>

AUTHSERVER <" " | '*domain-name*' | '.'>

Optional Arguments

" "
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: Windows" on page 592

AUTOEXEC System Option: Windows

Specifies the SAS autoexec file.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	AUTOEXEC.SAS, if the file is available; otherwise, none
Windows specifics:	*file-specification* must be a valid Windows file name

Syntax

-AUTOEXEC *file-specification*

-NOAUTOEXEC

Required Arguments

AUTOEXEC ***(file-specification1<...file-specification-n>)***
specifies the SAS autoexec file to be used instead of the default AUTOEXEC.SAS file. The *file-specification* argument can be a valid Windows file name or an environment variable associated with a pathname. For more information about the SAS autoexec file, see "SAS Autoexec File" in the "Getting Started" section in *SAS Companion for Windows*.

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 libraries that you access routinely in SAS sessions.

If no AUTOEXEC.SAS file is found, the default value for this option is NOAUTOEXEC.

You can use the APPEND and INSERT system options to add additional file specifications.

See Also

- "SAS Autoexec File " on page 34
- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

AWSCONTROL System Option: Windows

Specifies whether the main SAS window includes a title bar, a system control menu, and minimize maximize buttons.

Valid in: configuration file, SAS invocation, OPTIONS statement, System Options window

Category: Environment Control: Display

PROC OPTIONS GROUP= ENVDISPLAY

Default: TITLE SYSTEMMENU MINMAX

Windows specifics: all

Syntax

-AWSCONTROL <TITLE | NOTITLE> <SYSTEMMENU | NOSYSTEMMENU> <MINMAX | NOMINMAX>

AWSCONTROL= <TITLE | NOTITLE> <SYSTEMMENU | NOSYSTEMMENU> <MINMAX | NOMINMAX>

Optional Arguments

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 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 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 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: Windows" on page 676

AWSDEF System Option: Windows

Specifies the location and dimensions of the main SAS window when SAS initializes.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	80% of the display height and width
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*

Required Arguments

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 values, 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 values, the SAS window occupies your entire display. If you specify 50 for each of these values, 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 92.

AWSMENU System Option: Windows

Specifies whether to display the menu bar in the main SAS window.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	AWSMENU
Windows specifics:	all

Syntax

-AWSMENU | **-NOAWSMENU**

AWSMENU | **NOAWSMENU**

Required Arguments

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: Windows

Specifies whether to embed menu items that are specific to Windows in the main menus.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	AWSMENUMERGE
Windows specifics:	all

Syntax

-AWSMENUMERGE | **-NOAWSMENUMERGE**

AWSMENUMERGE | **NOAWSMENUMERGE**

Syntax Description

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: Windows" on page 716

AWSTITLE System Option: Windows

Replaces the default text in the main SAS title bar.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display

PROC OPTIONS GROUP= ENVDISPLAY

Default: none

Windows specifics: all

Syntax

-AWSTITLE "*title-text*"

Required Argument

"*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 enables 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: Windows

Specifies the number of buffers to be allocated for processing SAS data sets.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES, PERFORMANCE

Default: 1

Windows specifics: Default value

See: "BUFNO= System Option" in *SAS System Options: Reference*

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

System Arguments

***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: Windows" on page 601
- Chapter 3, "Using the SAS Editors under Windows," on page 107

BUFSIZE System Option: Windows

Specifies the permanent buffer page size for output SAS data sets.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Files: SAS Files
PROC OPTIONS GROUP=	SASFILES, PERFORMANCE
Default:	0
Windows specifics:	Valid values for *n*
See:	"BUFNO= System Option" in *SAS System Options: Reference*

Syntax

-BUFSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MAX

BUFSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MAX

Required Arguments

***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.

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 select a value depending on the size of the observation.

Experienced users might 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: Windows" on page 599

CATCACHE System Option: Windows

Specifies the number of SAS catalogs to keep open.

Valid in:	configuration file, SAS invocation
Category:	Files: SAS Files
PROC OPTIONS GROUP=	SASFILES
Default:	0
Windows specifics:	Valid values for *n*
See:	"CATCACHE= System Option" in *SAS System Options: Reference*

Syntax

-CATCACHE *n* | *n*K | MIN | MAX

Required Arguments

***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.

CLEANUP System Option: Windows

Specifies how to handle an out-of-resource condition.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Error Handling
PROC OPTIONS GROUP=	ERRORHANDLING
Default:	CLEANUP
Windows specifics:	behavior when running in batch mode
See:	"CLEANUP System Option" in *SAS System Options: Reference*

Syntax

-CLEANUP | **-NOCLEANUP**

CLEANUP | **NOCLEANUP**

Required Arguments

CLEANUP
specifies that during the entire session, SAS attempts to perform automatic, continuous cleanup of resources that are not essential for execution. Nonessential resources include those resources that are not visible to the user (for example, cache memory) and 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: Windows

Specifies the location where the SAS Command window is displayed.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	BOTTOM CENTER
Windows specifics:	all

Syntax

-COMDEF TOP | CENTER | BOTTOM
<LEFT | CENTER | RIGHT>

Syntax Description

TOP | CENTER | BOTTOM
specifies the vertical position of the Command window. The default value is BOTTOM.

LEFT | CENTER | RIGHT
specifies the horizontal position of the 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 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 84
- "Using the Command Bar to Issue Commands" on page 63

CONFIG System Option: Windows

Specifies the configuration file that is used when initializing or overriding the values of SAS system options.

Valid in:	configuration file, SAS invocation
Category:	System Administration: Installation
PROC OPTIONS GROUP=	INSTALL
Default:	!*sasroot*\SASV9.CFG
Windows specifics:	all

Syntax

-CONFIG *file-specification*

Required Argument

file-specification
specifies the file name of the SAS configuration file that you want to use, or a Windows environment variable that resolves to a valid file name. The *file-specification* must be a valid Windows file name. If *file-specification* contains spaces, it must be enclosed in quotation marks.

Details

The CONFIG system option specifies the complete file name 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 that you choose.

See Also

"SAS Configuration Files " on page 28

DEVICE System Option: Windows

Specifies a device driver for graphics output for SAS/GRAPH software.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Graphics: Driver Settings
PROC OPTIONS GROUP=	GRAPHICS
Alias:	-DEV
Default:	none
Windows specifics:	Valid values for *device-driver-name*; default value
See:	"DEVICE= System Option" in *SAS/GRAPH: Reference*

Syntax

-DEVICE *device-driver-name*

DEVICE= *device-driver-name*

Required Argument

device-driver-name
specifies the name of a device driver for graphics output.

Restriction The maximum number of characters for the device name is 8 characters.

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 that are referenced by the GDEVICE0 libref. See your on–site SAS support personnel for more information.

ECHO System Option: Windows

Specifies a message to be echoed to the SAS log while initializing SAS.

Valid in:	configuration file, SAS invocation
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOECHO
Windows specifics:	all

Syntax

-ECHO "*message*" | -NOECHO

Required Arguments

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: 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 System Options: Reference*
- "The SAS Log" in *SAS Programmer's Guide: Essentials*

EMAILDLG System Option: Windows

Specifies whether to use the native email dialog box provided by your email application or the email dialog box provided by SAS.

Valid in:	configuration file, SAS invocation
Category:	Communications: E-mail
PROC OPTIONS GROUP=	E-MAIL
Default:	NATIVE
Windows specifics:	all

Syntax

-EMAILDLG NATIVE | SAS

Required Arguments

NATIVE
specifies to use the email dialog box provided by your email system vendor. You can use the native dialog box with SAS only if the email system supports the MAPI interface.

SAS
specifies to use the email dialog box provided by SAS.

Details

The EMAILDLG system option specifies whether to use the native email interactive dialog box provided by your email application or the email interface provided by SAS. SAS uses the native dialog box by default.

See Also

"Sending Email Using SAS" on page 65

EMAILSYS System Option: Windows

Specifies the email protocol to use for sending electronic mail.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Communications: E-mail
PROC OPTIONS GROUP=	EMAIL
Default:	MAPI
Windows specifics:	all

Syntax

-EMAILSYS MAPI | VIM | SMTP

Required Arguments

MAPI
specifies to use the Messaging Application Programming Interface (MAPI) electronic mail interface. This value is the default.

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 protocols: MAPI (such as Microsoft Exchange), Vendor Independent Mail (VIM) 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 email programmatically. SMTP is not available using either your email program native dialog box or the SAS email dialog box.

See Also

- "Sending Email Using SAS" on page 65
- *SAS System Options: Reference*
- EMAILID System Option
- EMAILPW System Option
- EMAILAUTHPROTOCOL System Option
- "How the SMTP e-Mail Interface Authenticates Users" in *SAS Programmer's Guide: Essentials*

ENGINE System Option: Windows

Specifies the default access method to use for SAS libraries.

Valid in:	configuration file, SAS invocation
Category:	Files: SAS Files
PROC OPTIONS GROUP=	SASFILES
Default:	V9
Windows specifics:	valid values

Syntax

-ENGINE *engine-name*

Required Argument

engine-name
can be one of the following under Windows:

V9
specifies the default SAS engine for SAS 9 files.

Alias BASE

SAS/ACCESS engines (multiple)
specifies the engine for files from other applications.

SPDE
specifies the engine for SPD Engine files.

CAS
specifies the engine for in-memory tables on the SAS Cloud Analytic Services (CAS) server.

CVP
expands the length of character variables to avoid truncation when a data set is transcoded.

JSON
specifies the engine for JSON data.

XMLV2
XML
specifies the default engine for XML 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 for 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.

XPORT
specifies the transport engine.

Details

The default engine is used when a SAS 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

- Chapter 4, "Using SAS Files under Windows," on page 151
- "SAS Engines" in *SAS Programmer's Guide: Essentials*
- *SAS V9 LIBNAME Engine: Reference*
- *SAS/ACCESS for Relational Databases: Reference*

ENHANCEDEDITOR System Option: Windows

Specifies whether to enable the Enhanced Editor during SAS invocation.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	ENHANCEDEDITOR
Windows specifics:	all

Syntax

-ENHANCEDEDITOR | **-NOENHANCEDEDITOR**

Required Arguments

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: Windows" on page 427

FILELOCKWAIT System Option: Windows

Sets the number of seconds that SAS waits for a locked file to become available.

Valid in:	configuration file, SAS invocation
Category:	SAS Files
PROC OPTIONS GROUP=	SASFILES
Default:	0
Restriction:	FILELOCKWAIT= does not apply to SPD Engine files.
Interaction:	FILELOCKWAIT= is both a system option and a LIBNAME option. The system option applies to all SAS I/O files (libraries, CATALOG, DATA, VIEW, and MDDB) and external files. The LIBNAME option applies to the members in a library only. The LIBNAME option overrides the system option.
Windows specifics:	all

Syntax

FILELOCKWAIT= *wait-time*

Required Argument

wait-time
specifies the amount of time, in seconds, that SAS waits for a locked file to become available.

Details

Normally, SAS returns an error if the file that it attempts to access is locked. With the FILELOCKWAIT= system option, you can limit or turn on the amount of time SAS waits for a locked file to become available. When you set FILELOCKWAIT= to a value of *wait-time*, SAS waits the specified amount of time for the file to become available before failing. When the time limit is reached, SAS returns a locked-file error and the DATA step fails. The maximum time that you can set to wait for a locked file is 10 minutes. When you set FILELOCKWAIT= to 0, SAS immediately fails.

SAS continues to check the availability of the file based on the time value specified. For example, if FILELOCKWAIT=10, then SAS continues to check if the file is available for 10 sec. If after 10 secs, the file is still not available, then an error message is displayed to the SAS log.

The FILELOCKWAIT system option is used primarily by a system administrator, who can change the default value of FILELOCKWAIT= by using the FILELOCKWAITMAX system option. This option can also be restricted by a system administrator.

You can also use the FILELOCKWAIT= statement option in the LIBNAME statement to set a time limit for the files in your library.

FILELOCKWAITMAX= System Option: Windows

Sets an upper limit on the amount of time that SAS waits for a locked file.

Valid in:	configuration file, SAS invocation
Category:	Files: SAS Files
PROC OPTIONS GROUP=	SASFILES
Default:	600
Windows specifics:	all

Syntax

FILELOCKWAITMAX= *wait-time*

Required Argument

wait-time
specifies the amount of time, in seconds, that SAS waits for a locked file to become available.

Default 600

Range 0 - 600

Interactions Specifying the FILELOCKWAITMAX= system option can have an adverse effect on one or more SAS/SHARE server and client sessions that are waiting for the release of a SAS file that is locked by another process. One or more wait conditions could lead to failed processes for a SAS/SHARE server and clients.

To prevent the possibility of a failed SAS/SHARE process, you can specify FILELOCKWAITMAX=0, which cancels the amount of time that a SAS/SHARE server and clients would wait for the release of a locked file. Canceling the wait time would prevent a failed process.

Details

The FILELOCKWAITMAX= system option enables you to limit or turn off the amount of time SAS waits for a locked file. SAS uses the FILELOCKWAIT= LIBNAME option to wait for the file to become available. Using the FILELOCKWAITMAX= system option, an administrator can limit or turn off this behavior. Normally, SAS returns an error if the file that it attempts to access is locked. If you set FILELOCKWAITMAX= to 0, SAS fails immediately after encountering a locked file. This option is used primarily by a system administrator.

See Also

"LIBNAME Statement: Windows" on page 556

FILTERLIST System Option: Windows

Specifies an alternative set of file filter specifications to use for the Open and Save As dialog boxes.

Valid in: configuration file, SAS invocation

Category: Environment Control: Display

PROC OPTIONS GROUP= ENVDISPLAY

Default: none

Windows specifics: all

Syntax

-FILTERLIST "*filter1* | *filter2*|... | *filter-n*"

Required Argument

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 file 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** dialog 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: Windows" on page 394
- "DLGSAVE Command: Windows" on page 401

FONT System Option: Windows

Specifies a font to use for SAS windows.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	Sasfont 8
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*>

Required Arguments

"font-name"
specifies the name of the font for text in the SAS windowing environment. This name must be a valid font name (for example, "SAS Monospace" or "Courier"). The *font-name* argument must be enclosed in double quotation marks. This argument is required.

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 value 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 that 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 Fonts 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 can display incorrectly. If you specify a *point-size* that is not valid for a font, SAS uses the closest point size for the font that you specify.

See Also

- "SYSGUIFONT System Option: Windows" on page 700
- "SYSPRINTFONT System Option: Windows" on page 705

FONTALIAS System Option: Windows

Assigns a Windows font to one of the SAS fonts.

Valid in:	configuration file, SAS invocation
Category:	Graphics: Driver Settings
PROC OPTIONS GROUP=	GRAPHICS
Default:	varies (see table in the Details section.)
Windows specifics:	all

Syntax

-FONTALIAS "*SAS-font*" "*host-specific-font*"

Required Arguments

"*SAS-font*"
specifies the SAS font that 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:

Table 21.2 *Default Font Aliases*

SAS font	Windows font
Times	Times New Roman
Helvetica	Arial
Courier	Courier New

SAS font	Windows font
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: Windows

Specifies the location of the SAS fonts that are loaded during the SAS session.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY ODSPRINT
Default:	!*sasroot*\core\resource
Windows specifics:	all
See:	"FONTSLOC= System Option" in *SAS System Options: Reference*

Syntax

-FONTSLOC *directory-specification*

Required Argument

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: Windows

Specifies the default output formatting characters.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: Procedure Output
PROC OPTIONS GROUP=	LISTCONTROL
Default:	(see the SAS configuration file)
Windows specifics:	Valid values for *formatting-characters*
See:	"FORMCHAR= System Option" in *SAS System Options: Reference*

Syntax

-FORMCHAR "*formatting-characters*"

FORMCHAR="*formatting-characters*"

Required Argument

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 hexadecimal constant with this option, SAS interprets the value of the hexadecimal constant as appropriate for the Windows environment.

The configuration file shipped with SAS contains two FORMCHAR system option specifications. One is 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.

Note: To ensure that row and column separators and boxed tabular reports are printed legibly when using the standard forms characters, you must use these resources:

- the SAS Monospace or the SAS Monospace Bold font
- a printer that supports TrueType fonts

FULLSTIMER System Option: Windows

Specifies whether to write all available system performance statistics to the SAS log.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOFULLSTIMER
Windows specifics:	all

Syntax

-FULLSTIMER | **-NOFULLSTIMER**

FULLSTIMER | **NOFULLSTIMER**

Required Arguments

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. NOFULLSTIMER 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.

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 to specify the destination for a copy of the SAS log. For more information, see "ALTLOG System Option: Windows" on page 588. 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.

Here 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
      OS Memory           4648k
```

FULLSTIMER displays the following statistics:

Table 21.3 *Description of FULLSTIMER Statistics*

Statistic	Description
Real Time	the amount of time spent to process the SAS job. Real time is also referred to as elapsed time.
User CPU Time	the CPU time spent to execute SAS code.
System CPU Time	the CPU time spent to perform operating system tasks (system overhead tasks) that support the execution of SAS code

Statistic	Description
Memory	the amount of memory required to run a step.
OS Memory	the maximum amount of memory that a step requested from the System.

Note: Starting in SAS 9, some procedures use multiple threads. On computers with multiple CPUs, the operating system can run more than one thread simultaneously. Consequently, CPU time might exceed real time in your FULLSTIMER output.

For example, a SAS procedure could use two threads that run on two separate CPUs simultaneously. The value of CPU time would be calculated as the following:

```
CPU1 time + CPU2 time = total CPU time
1 second + 1 second = 2 seconds
```

Since CPU1 can run a thread at the same time that CPU2 runs a separate thread for the same SAS process, you can theoretically consume 2 CPU seconds in 1 second of real time.

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: Windows" on page 699
- "Optimizing System Performance" in *SAS Programmer's Guide: Essentials*
- "The SAS Log" in *SAS Programmer's Guide: Essentials*

HELPHOST System Option: Windows

Specifies the name of the computer where the remote browsing system is to be displayed.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Default: NULL

See: "HELPHOST System Option" in *SAS System Options: Reference*

Comparisons

The value of HELPHOST is based on the address of the computer running the remote desktop client. If you are logged in to a host, then you receive an error message.

HELPINDEX System Option: Windows

Specifies one or more index files for the SAS Help and Documentation.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Help
PROC OPTIONS GROUP=	HELP
Default:	/help/common.hlp/index.txt, /help/common.hlp/keywords.htm, common.hhk
Windows specifics:	*HTML-HELP-index-pathname*

Syntax

-HELPINDEX <(> "*index-pathname-1*" < " *index-pathname-2*" " *index-pathname-n*")>

Syntax Description

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: Windows" on page 625

HELPLOC System Option: Windows

Specifies the location of Help files that are used to view SAS Help and Documentation using Microsoft HTML Help.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Help
PROC OPTIONS GROUP=	HELP
Default:	("!MYSASFILES\classdoc" "!*sasroot*\nls\en\help" "!*sasroot*\core\help")
Windows specifics:	valid values for *pathname*

Syntax

-HELPLOC <(> "*pathname-1*" <"*pathname-2*" "*pathname-n*")>

Syntax Description

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 HELPLOC Value

Specifying a value for the HELPLOC system option causes SAS to insert that value at the start of a concatenated list of values. This action 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.

You can use the APPEND and INSERT system options to add additional file specifications.

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")
```

See Also

- “SAS Autoexec File ” on page 34
- “APPEND= System Option” in *SAS System Options: Reference*
- “INSERT= System Option” in *SAS System Options: Reference*

HELPREGISTER System Option: Windows

Registers help files to access from the main SAS window Help menu.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Help
PROC OPTIONS GROUP=	HELP
Default:	none
Windows specifics:	all

Syntax

-HELPREGISTER "*menu string*" *help file location* <"*help string*"> <*topic*> <CHM | HLP | HTML>

Syntax Description

"*menu string*"
is the text string that appears in the Help menu.

help file location
specifies the folder and the file name 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* can 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 bar when a user positions the pointer over the *menu string*.

topic
is the topic within the Help file that is displayed when you select *menu string* from SAS help menu. For HTML files, the topic is the anchor (preceded by #) 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 can 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”
```

HELPTOC System Option: Windows

Specifies the table of contents files for the SAS Help and Documentation.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Help
PROC OPTIONS GROUP=	HELP
Default:	/help/helpnav.hlp/config.txt /help/common.hlp/toc.htm common.hhc
Windows specifics:	*HTML-Help-TOC-pathname*

Syntax

-HELPTOC <(> "*TOC-pathname-1*" <"*TOC-pathname-2*" " *TOC-pathname-3*")>

Syntax Description

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 pathname 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 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 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: Windows" on page 625

HOSTINFOLONG System Option: Windows

Specifies to write additional operating-environment information in the SAS log when SAS starts.

Valid in:	Configuration file, SAS invocation
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL

Syntax

HOSTINFOLONG | **NOHOSTINFOLONG**

Syntax Description

HOSTINFOLONG
specifies to print additional operating-environment information in the SAS log when SAS starts.

NOHOSTINFOLONG
specifies to omit additional operating environment information in the SAS log when SAS starts.

Details

When HOSTINFOLONG is specified at start-up, SAS writes additional information about the operating environment to the SAS log. The system default is HOSTINFOLONG. Here is an example:

```
NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.4 (TS1M0)
      Licensed to SAS Institute Inc., Site 1.
NOTE: This session is executing on the W32_7PRO  platform.



NOTE: Updated analytical products:

      SAS/STAT 12.3 (maintenance)
      SAS/ETS 12.3 (maintenance)
      SAS/OR 12.3 (maintenance)
      SAS/IML 12.3 (maintenance)
      SAS/QC 12.3 (maintenance)

NOTE: Additional host information:

 W32_7PRO DNTHOST 6.1.7601 Service Pack 1 Workstation

NOTE: SAS initialization used:
      real time           1.42 seconds
      cpu time            0.70 seconds
```

HOSTPRINT System Option: Windows

Specifies that the Windows Print Manager is to be used for printing.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: Procedure Output
PROC OPTIONS GROUP=	LISTCONTROL
Default:	HOSTPRINT
Windows specifics:	all

Syntax

-HOSTPRINT | **-NOHOSTPRINT**

HOSTPRINT | **NOHOSTPRINT**

Syntax Description

HOSTPRINT
specifies to use Windows printing. HOSTPRINT 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 line size, page size, and font values that are specified in your SAS form.

See Also

"Setting Print Options to Use Forms" on page 220

ICON System Option: Windows

Minimizes the SAS window.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Option window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	NOICON
Windows specifics:	all

Syntax

-ICON | **-NOICON**

ICON | **NOICON**

Syntax Description

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 action 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: Windows

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
Category:	Environment Control: Initialization and Operation
PROC OPTIONS GROUP=	EXECMODES
Alias:	IS
Default:	none
Windows specifics:	*statement* must end a DATA or PROC step if you use the Enhanced Editor
See:	"INITSTMT= System Option" in *SAS System Options: Reference*

Syntax

INITSTMT '*statement*'

Required Argument

'statement'
specifies any SAS statement or statements. The value of *statement* must end a DATA or PROC step if you use the Enhanced Editor.

INSERT System Option: Windows

Used when SAS starts, inserts the specified value at the beginning of the specified system option.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
PROC OPTIONS GROUP=	ENVFILES
Default:	none
Windows specifics:	SAS invocation syntax

Syntax

-INSERT *system-option argument*

Required Arguments

system-option
can be AUTOEXEC, CMPLIB, FMTSEARCH, HELPLOC, MAPS, MSG, SASSCRIPT, SASAUTOS, SASHELP, or SET.

argument
specifies a new pathname or an environment setting that you want to insert at the front of the current value of *system-option*. The following example shows that a library is being inserted before the FMTSEARCH option:

```
-set APFMTLIB "SASEnvironment/SASFormats"
-insert fmtsearch APFMTLIB
```

Details

By default, if you specify the AUTOEXEC, CMPLIB, FMTSEARCH, HELPLOC, MAPS, MSG, SASSCRIPT, SASAUTOS, SASHELP, or SET system option more than one time, the last specification is the one that SAS uses. If you want to insert additional pathnames in front of the search paths that have already been specified by one of these options, you must use the INSERT or APPEND system options to append the new pathname. For example, if you entered the following SAS command, the only location that SAS looks for help files is **c:\app2\help,** and the output of PROC OPTIONS shows **c:\app2\help**:

```
sas -helploc "c:\app2\help"
```

If you want SAS to look in both the current path for help files and in **c:\app2\help** and if you want SAS to search **c:\app2\help** first, then you can append the new location to the value of the HELPLOC option using the INSERT option:

```
sas -insert helploc "c:\app2\help"
```

If your current path for help files is **!sasroot\nls\en\help**, then for the value of the HELPLOC option, PROC OPTIONS shows

```
("c:\app2\help" "!sasroot\nls\en\help")
```

See Also

- "Changing an Option Value By Using the INSERT and APPEND System Options" in *SAS System Options: Reference*
- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

JREOPTIONS System Option: Windows

Identifies Java Runtime Environment (JRE) options for SAS.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Initialization and Operation
PROC OPTIONS GROUP=	EXECMODES
Default:	-Djava.security.policy=<*pathname**filename*> -Dsas.jre=(private | public) -Dsas.jre.home=!*sasroot**pathname* -Djava.ext.dirs=*pathname**filename*
Windows specifics:	all
Important:	Djava.security.manager should be added to JREOPTIONS in order to mitigate a security risk.
CAUTION:	**Before you change the settings for the JREOPTIONS option, contact SAS Technical Support to make sure that the Java setting that you want to change does not cause SAS to crash. A best practice is to change only the Java properties for your own Java code.**

Syntax

-JREOPTIONS (-*JRE-option-1* <-*JRE-option-n*>)

JREOPTIONS (-*JRE-option-1* <-*JRE-option-n*>)

Syntax Description

-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.

Example: Examples

- `-jreoptions \(-Dmy.java.property\)`
- `-jreoptions \(-Xmx512m -Xms256m\)`

LINESIZE System Option: Windows

Specifies the line size of SAS Log and Output windows.

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
Default:	Varies depending on display settings
Windows specifics:	Default value
See:	"LINESIZE= System Option" in *SAS System Options: Reference*

Syntax

-LINESIZE *n* | MIN | MAX

LINESIZE= *n* | MIN | MAX

Required Arguments

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 are printed 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: Windows" on page 657
- "ORIENTATION= System Option" in *SAS System Options: Reference*
- "PAPERSIZE= System Option" in *SAS System Options: Reference*

LOADMEMSIZE System Option: Windows

Specifies a suggested amount of memory needed for executable programs loaded by SAS.

Valid in:	configuration file, SAS invocation
Category:	System Administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	0
Windows specifics:	all

Syntax

-LOADMEMSIZE*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***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.

LOG System Option: Windows

Specifies a destination for a copy of the SAS log when SAS is running in batch mode.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES, LOGCONTROL
Default:	*filename*.LOG in batch mode, where *filename* is the name of your SAS job
Windows specifics:	*destination* must be a valid Windows file name

Syntax

-LOG "*destination*" | -NOLOG

Required Arguments

LOG "*destination*"
specifies the destination for the SAS log. The *destination* argument can be a valid Windows pathname or file name (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 for a copy of the SAS log when running 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, the log 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.

When SAS is started with the OBJECTSERVER and NOTERMINAL system options active, and no log is specified, SAS discards all log and alternate log messages.

Using directives in the value of the LOG system option enables you to control when logs are open and closed and how they are named, based on real-time events, such as time, month, day of week. For a list of directives see the LOGPARM= system option in the *SAS System Options: Reference*.

If you start SAS in batch mode or in server mode and if the LOGCONFIGLOC= option is specified, logging is done by the SAS logging facility. The traditional SAS log option LOGPARM= is ignored. The traditional SAS log option LOG= is applied only when the %S{App.Log} conversion character is specified in the logging configuration file. For more information, see *SAS Logging: Configuration and Programming Reference*.

See Also

- "TERMINAL System Option" in *SAS System Options: Reference*
- "LOGPARM= System Option" in *SAS System Options: Reference*
- "The SAS Log" in *SAS Programmer's Guide: Essentials*

MAPS System Option: Windows

Specifies the name of the SAS library that holds the SAS/GRAPH map data sets.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Graphics: Driver Settings
PROC OPTIONS GROUP=	GRAPHICS
Default:	!*sasroot*\maps
Windows specifics:	default value and *location-of-maps* must resolve to a valid Windows pathname
See:	"MAPS= System Option" in *SAS/GRAPH and Base SAS: Mapping Reference*

Syntax

-MAPS *location-of-maps*

MAPS=*location-of-maps*

Required Argument

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.

Details

APPEND and INSERT System Options

You can use the APPEND and INSERT system options to add additional file specifications.

See Also

- "SAS Autoexec File " on page 34
- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

MAXMEMQUERY System Option: Windows

Specifies the limit on the maximum amount of memory that is allocated for procedures.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	System Administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	0
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

Note: You can also use the KB, MB, GB, and TB syntax notations.

Required Arguments

***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 value 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 use the MAXMEMQUERY option to specify the largest block of virtual memory that SAS can request at one time. By contrast, the MEMSIZE option places a limit on the total amount of virtual memory that SAS dynamically allocates at any time. This virtual memory is supported by a combination of real memory and paging space. The operating environment begins paging when the amount of virtual memory that is required exceeds the real memory that is available. To prevent paging and the associated performance problems, the MAXMEMQUERY and MEMSIZE system options should be set to a subset of real memory.

MEMBLKSZ System Option: Windows

Specifies the memory block size for memory-based libraries for Windows operating environments.

Valid in:	configuration file, SAS invocation
Category:	System Administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	16M
Windows specifics:	all

Syntax

-MEMBLKSZ *n* | *n*K | *n*M | *n*G | *n*T | *hex*X

Required Arguments

***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:

- 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 might 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 249
- “MEMMAXSZ System Option: Windows” on page 645

MEMCACHE System Option: Windows

Specifies to use the memory-based libraries as a SAS file cache.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: SAS Files

PROC OPTIONS GROUP=	SASFILES
Default:	0
Windows specifics:	all

Syntax

-MEMCACHE 0 | 1 | 4

MEMCACHE= 0 | 1 | 4

Required Arguments

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 249
- “MEMLIB System Option: Windows” on page 644

MEMLIB System Option: Windows

Specifies to process the Work library as a memory-based library.

Valid in:	configuration file, SAS invocation
Category:	Files: SAS Files
PROC OPTIONS GROUP=	SASFILES
Default:	NOMEMLIB
Windows specifics:	all

Syntax

-MEMLIB | **-NOMEMLIB**

Required Arguments

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 249
- “LIBNAME Statement: Windows” on page 556
- “MEMCACHE System Option: Windows” on page 643
- “Performance Tools” on page 267

MEMMAXSZ System Option: Windows

Specifies the maximum amount of memory to allocate for using memory-based libraries in Windows operating environments.

Valid in:	configuration file, SAS invocation

Category:	System Administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	2G
Windows specifics:	all

Syntax

-MEMMAXSZ *n* | *n*K | *n*M | *n*G | *n*T | *hex*X

Required Arguments

***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.

CAUTION
Specifying a value that is too large can 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 249
- “MEMBLKSZ System Option: Windows” on page 642
- “MEMLIB System Option: Windows” on page 644

■ “MEMCACHE System Option: Windows” on page 643

MEMSIZE System Option: Windows

Specifies the limit on the amount of virtual memory that can be used during a SAS session.

Valid in:	configuration file, SAS invocation
Category:	System Administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	2G
Windows specifics:	valid values

Syntax

-MEMSIZE *n* | *n*K | *n*M | *n*G | *n*T | *hex*X | MAX

Note: You can also use the KB, MB, GB, and TB syntax notations.

Required Arguments

***n* | *n*K | *n*M | *n*G | *n*T**
specifies the limit in bytes, kilobytes (1024 bytes), megabytes (1,048,576 bytes), gigabytes (1,073,741,824 bytes), or terabytes (1,099,511,627,776 bytes). For example, a value of `0.25G` is equivalent to 268,435,456 bytes, and 16.5M is equivalent to 17,301,504 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 hexadecimal characters (0–9, A-F), and then followed by an X. For example, `0F00000x` sets the value of the MEMSIZE option to 15,728,640 bytes, which is equivalent to a value of `0x`

MAX
specifies the largest reasonable value dependent on the amount of physical memory and paging space available when SAS is started.

Details

The MEMSIZE system option specifies the total amount of memory available to each SAS session. A value that is too low results in out-of-memory conditions.

A numeric value of 0 (or 0x) is equivalent to the option value MAX.

If an unreasonably small numeric value is specified (for example, 6K), the setting of the MEMSIZE option is silently increased to a minimum reasonable value that allows SAS to start and have basic functionality.

If a numeric value greater than 4,294,967,295 is specified on a 32-bit version of SAS, the setting of the value is silently reduced to 4,294,967,295.

Numeric values greater than 9,223,372,036,854,775,807 bytes is rejected as invalid, and prevents SAS from starting.

SAS does not automatically reserve or allocate the amount of memory that you specify in the MEMSIZE system option. SAS uses only as much memory as it needs to complete a process. For example, a DATA step might require only 20M of memory, so even though MEMSIZE is set to 500M, SAS uses only 20M of memory.

While your SAS jobs are running, you can monitor the effect of larger memory settings by using system monitoring tools.

Note: Setting MEMSIZE to MAX is reasonable only if consumers of large amounts of memory are not likely to become active after SAS has started. For example, if multiple instances of SAS are running concurrently, and all of these instances are started with a MEMSIZE value of MAX, one or more of these instances can encounter out of memory conditions, or, the operating system can run out of available paging space.

MSG System Option: Windows

Specifies the library that contains the SAS error messages.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	!*sasroot*\core\sasmsg
Windows specifics:	Valid values for *library-specification*

Syntax

-MSG *library-specification*

Required Argument

library-specification
can be a Windows logical name (including search strings) or pathname. Do not include a file name. 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.

You can use the APPEND and INSERT system options to add additional file specifications.

See Also

- "SAS Autoexec File " on page 34
- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

MSGCASE System Option: Windows

Specifies whether notes, warnings, and error messages that are generated by SAS are displayed in uppercase characters.

Valid in:	configuration file, SAS invocation
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOMSGCASE
Windows specifics:	all

Syntax

-MSGCASE | **-NOMSGCASE**

Syntax Description

MSGCASE
specifies that messages are displayed in uppercase characters.

NOMSGCASE
specifies that messages can include uppercase and lowercase characters. NOMSGCASE is the default.

Details

Specifies whether notes, warnings, and error messages that are generated by SAS are displayed in uppercase characters. The setting of the MSGCASE option does not affect user-generated messages and source lines.

MSGCASE is supported in the NL formats. For information about the NL formats, see the *SAS National Language Support (NLS): Reference Guide*.

See Also

"The SAS Log" in *SAS Programmer's Guide: Essentials*

MSYMTABMAX System Option: Windows

Specifies the maximum amount of memory available to the macro variable symbol table(s).

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Macro: SAS Macro
PROC OPTIONS GROUP=	MACRO
Default:	4194304 bytes (4 MB)
Windows specifics:	Default value
See:	MSYMTABMAX 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

Required Arguments

***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 value 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: Windows

Specifies the maximum size for in-memory macro variables.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Macro: SAS Macro
PROC OPTIONS GROUP=	MACRO
Default:	65534 bytes
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

Note: You can also use the KB, MB, and GB syntax notation.

Required Arguments

***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 value 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: Windows

Specifies a file that contains messages to be written to the SAS log.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
Windows specifics:	Valid values for *file-specification*
See:	"NEWS= System Option" in *SAS System Options: Reference*

Syntax

-NEWS *file-specification*

Required Argument

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.

NUMKEYS System Option: Windows

Controls the number of available function keys.

Valid in:	configuration file, SAS invocation
Category:	Input Control: Data Processing
PROC OPTIONS GROUP=	INPUTCONTROL
Default:	number of function keys on the keyboard
Windows specifics:	all

Syntax

-NUMKEYS *number-of-keys*

Required Argument

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: Windows

Specifies the number of mouse buttons SAS is displayed in the KEYS window.

Valid in:	configuration file, SAS invocation
Category:	Input Control: Data Processing
PROC OPTIONS GROUP=	INPUTCONTROL
Default:	3 buttons
Windows specifics:	all

Syntax

-NUMMOUSEKEYS *number-of-buttons*

Required Argument

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 configuration, specify the NUMMOUSEKEYS system option in your SAS configuration file.

OBS System Option: Windows

Specifies when to stop processing observations or records.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Files: SAS Files
PROC OPTIONS GROUP=	SASFILES
Default:	MAX
Windows specifics:	Valid range
See:	"OBS= System Option" in *SAS System Options: Reference*

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

Required Arguments

***n* | *n*K | *n*M | *n*G | *n*T**

specifies a number to indicate when to stop processing, with *n* as 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.

OPLIST System Option: Windows

Specifies whether the settings of the SAS system options are written to the SAS log.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOOPLIST
Windows specifics:	all

Syntax

-OPLIST | **-NOOPLIST**

Details

The OPLIST system option echoes only the system options specified on the command line; it does not echo any system options specified in the configuration file or in the SASV9_OPTIONS environment variable. (If you want to echo the contents of the configuration file, use the VERBOSE option.) For example, invoke SAS with the following command:`sas -nodms -fullstimer -nonews -oplist`. SAS writes this line to the SAS log: `NOTE: SAS command line: -nodms -fullstimer -nonews -oplist`.

PAGENO System Option: Windows

Resets the page number.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: Procedure Output

PROC OPTIONS GROUP= LISTCONTROL

Default: 1

Windows specifics: Valid values for *n*; syntax

See: "PAGENO= System Option" in *SAS System Options: Reference*

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

Required Arguments

***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: Windows

Specifies the number of lines that compose a page of SAS output.

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

Default: Varies depending on your display settings

Windows specifics: Default value

See: "PAPERSIZE= System Option" in *SAS System Options: Reference*

Syntax

-PAGESIZE *n* | MIN | MAX

PAGESIZE=*n* | MIN | MAX

Required Arguments

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 are printed 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 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

"LINESIZE System Option: Windows" on page 636

PAPERTYPE System Option: Windows

Specifies to a printer the type of paper to use for printing.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: ODS Printing
PROC OPTIONS GROUP=	ODSPRINT
Default:	PLAIN
Windows specifics:	valid values
See:	"PAPERTYPE= System Option" in *SAS System Options: Reference*

Syntax

-PAPERTYPE PLAIN | STANDARD | GLOSSY | TRANSPARENCY | *printer-defined-value*

PAPERTYPE=PLAIN | STANDARD | GLOSSY | TRANSPARENCY | *printer-defined-value*

Required Arguments

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.

PATH System Option: Windows

Specifies one or more search paths for SAS executable files.

Valid in:	configuration file, SAS invocation
Category:	System Administration: Installation
PROC OPTIONS GROUP=	INSTALL
Default:	!*sasroot*\core\sasexe
Windows specifics:	all

Syntax

-PATH <(>"*directory-specification-1*" <"*directory-specification-n*")>

Syntax Description

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

The PATH option identifies the search paths for SAS executable files. You can specify multiple PATH options to define the search order. The paths are searched in the order in which SAS encounters them. Therefore, specify at the front of the list the paths for the products that you run most frequently.

PFKEY System Option: Windows

Specifies which set of function keys to designate as the primary set of function keys.

Valid in:	configuration file, SAS invocation
Category:	Input Control: Data Processing

PROC OPTIONS GROUP= INPUTCONTROL

Default: WIN

Windows specifics: all

Syntax

-PFKEY PRIMARY | ALTERNATE | SAA | WIN

Required Arguments

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.

Here are the primary mainframe key definitions:

Table 21.4 *Primary Mainframe Key Definitions*

PC Key	Mainframe Definition	Key	Mainframe 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

PC Key	Mainframe Definition	Key	Mainframe Definition
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.

Here are the alternate mainframe key definitions:

Table 21.5 *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.

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.

Here are the IBM SAA key definitions:

Table 21.6 *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 entering `keys` in the command bar.

PRIMARYPROVIDERDOMAIN= System Option: Windows

Specifies the domain name of the primary authentication provider.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Initialization and Operation
PROC OPTIONS GROUP=	EXECMODES
Alias:	PRIMPD=
See:	PRIMARYPROVIDERDOMAIN= System Option

PRINT System Option: Windows

Specifies a destination for SAS output when running in batch mode.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	*filename*.LST in batch mode, where *filename* is the name of your SAS job
Windows specifics:	all

Syntax

-PRINT *destination* | -NOPRINT

Syntax Description

PRINT *destination*
specifies the destination for the SAS procedure output file. The *destination* argument can be a valid Windows pathname or file name (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.

PRNGETLIST System Option: Windows

Specifies whether printers that are attached to the system are recognized.

Valid in:	configuration file, SAS invocation
Categories:	Log and Procedure Output Control: Procedure Output Log and Procedure Output Control: ODS Printing
PROC OPTIONS GROUP=	LISTCONTROL ODSPRINT
Default:	PRNGETLIST
Windows specifics:	all

Syntax

PRNGETLIST | **NOPRNGETLIST**

Syntax Description

PRNGETLIST
specifies that SAS recognizes printers that are attached to the system

NOPRNGETLIST
specifies that SAS does not recognize printers that are attached to the system

Details

The PRNGETLIST option specifies that SAS recognizes all printers that are attached to the system. NOPRNGETLIST specifies that SAS does not recognize the printers that are attached to the system. NOPRNGETLIST can be used by SAS technical support as an alternative to advising users to delete all the printers on their system. NOPRNGETLIST can also be used when SAS is started by the object spawner in order to avoid the performance penalty of discovering printers and their capabilities

PRTABORTDLGS System Option: Windows

Specifies when to display the Print Abort dialog box.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	BOTH
Windows specifics:	all

Syntax

-PRTABORTDLGS BOTH | NEITHER | FILE | PRINTER

PRTABORTDLGS= BOTH | NEITHER | FILE | PRINTER

Required Arguments

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 224

PRTPERSISTDEFAULT System Option: Windows

Specifies to use the same destination printer from SAS session to SAS session.

Valid in:	configuration file, SAS invocation
Category:	Log and Procedure Output Control: ODS Printing
PROC OPTIONS GROUP=	ODSPRINT
Default:	NOPRTPERSISTDEFAULT
Windows specifics:	all

Syntax

-PRTPERSISTDEFAULT | **-NOPRTPERSISTDEFAULT**

Syntax Description

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 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: Windows" on page 703
- "Printing" on page 208

PRTSETFORMS System Option: Windows

Specifies whether to include the **Use Forms** check box in the Print Setup dialog box.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	PRTSETFORMS
Windows specifics:	all

Syntax

-PRTSETFORMS | **-NOPRTSETFORMS**

PRTSETFORMS | **NOPRTSETFORMS**

Syntax Description

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 220

REALMEMSIZE System Option: Windows

Specifies the amount of real memory SAS can expect to allocate.

Valid in:	configuration file, SAS invocation
Category:	System Administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	0
Windows specifics:	valid values

Syntax

-REALMEMSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Note: You can also use the KB, MB, and GB syntax notation.

Required Arguments

***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 that 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

Some SAS procedures use the REALMEMSIZE option to specify how much virtual memory the procedure can allocate and use without inducing excessive page swapping. By contrast, the MEMSIZE option places a limit on the total amount of virtual memory that SAS dynamically allocates at any time. This virtual memory is supported by a combination of real memory and paging space. The operating environment begins paging when the amount of virtual memory that is required exceeds the real memory that is available. To prevent paging and the associated performance problems, the REALMEMSIZE and MEMSIZE options should be set to a subset of real memory.

Comparisons

The REALMEMSIZE option is similar to the SORTSIZE option. The REALMEMSIZE option affects multiple procedures. The SORTSIZE option affects only the SORT procedure.

REGISTER System Option: Windows

Adds an application to the Tools menu in the main SAS window.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	none
Windows specifics:	all

Syntax

-REGISTER '*menu-name*' *'command'* <'*working-directory*'>

Syntax Description

'menu-name'
specifies the name that you want to appear in the menu. The *menu-name* must be enclosed in quotation marks.

'command'
specifies the command that 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 whether 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 menu in the main SAS window. If your menu name or command does not include blanks or special characters, you can omit the quotation marks. For more information about adding commands to the list, see "Adding Applications to the Tools Menu" on page 92 .

RESOURCESLOC System Option: Windows

Specifies a directory location of the files that contain SAS resources.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	!*sasroot*\core\resource
Windows specifics:	all

Syntax

-RESOURCESLOC <(>'*directory-specification-1*' <'*directory-specification-n*')> | "."

Syntax Description

'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: Windows

Controls whether members of the Sasuser data library can be opened for update or for Read-Only access.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	NORSASUSER
Windows specifics:	Network considerations
See:	“RSASUSER System Option” in *SAS System Options: Reference*

Syntax

-RSASUSER | **-NORSASUSER**

Syntax Description

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. The RSASUSER system option is extremely useful when users must share information (such as the Profile catalog) stored in the Sasuser data library. RSASUSER 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: Windows

Produces a list of resources that are read or loaded during a SAS session.

Valid in:	configuration file, SAS invocation
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NONE
Windows specifics:	all

Syntax

-RTRACE ALL | NONE

Required Arguments

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 RTRACELOC system options to create a file that lists the resources SAS uses. If you specify -RTRACE ALL but do not specify the RTRACELOC system option, the output is written to the SAS log.

See Also

- "RTRACELOC System Option: Windows" on page 674
- "The SAS Log" in *SAS Programmer's Guide: Essentials*

RTRACELOC System Option: Windows

Specifies the pathname of the file to which the list of resources that are read or loaded during a SAS session is written.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
Windows specifics:	all
Tip:	You can expand the RTRACELOC file name when %p (pid), %d (date), %t (time) are specified.

Syntax

-RTRACELOC *filename* | *pathname**filename*

Syntax Description

filename* | *pathname**filename
specifies a valid Windows file name or a pathname and a file name in which to store the file resource information. If the file name 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 RTRACE system options to determine which resources SAS uses.

See Also

"RTRACE System Option: Windows" on page 673

SASAUTOS System Option: Windows

Specifies the autocall macro library.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Categories:	Environment Control: Files Environment Control: SAS Macro
PROC OPTIONS GROUP=	ENVFILES; MACRO
Default:	SASAUTOS
Windows specifics:	Valid values for *library-specification*
See:	*SAS Macro Language: Reference*

Syntax

-SASAUTOS <(>*"library-specification-1"*...<*"library-specification-n"*)>

SASAUTOS=<(>*"library-specification-1"*...<*"library-specification-n"*)>

Syntax Description

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.

You can use the APPEND and INSERT system options to add additional file specifications.

See Also

- "SASAUTOS System Option" on page 735
- "SAS Autoexec File " on page 34
- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*
- *SAS Macro Language: Reference*.

SASCONTROL System Option: Windows

Specifies whether the SAS application windows include system and control menus and minimize and maximize buttons.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	SYSTEMMENU MINMAX
Windows specifics:	all

Syntax

-SASCONTROL SYSTEMMENU | NOSYSTEMMENU<MINMAX | NOMINMAX>

-SASCONTROL <SYSTEMMENU | NOSYSTEMMENU> MINMAX | NOMINMAX

SASCONTROL=SYSTEMMENU | NOSYSTEMMENU<MINMAX | NOMINMAX>

SASCONTROL=<SYSTEMMENU | NOSYSTEMMENU> MINMAX | NOMINMAX

Required Arguments

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: Windows" on page 594

SASHELP System Option: Windows

Specifies the directory or directories to be searched for SAS default forms, device lists, dictionaries, and other entries in the Sashelp catalog.

Valid in: configuration file, SAS invocation

Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	!*sasroot**SAS product*\sashelp, !*sascfg*\sascfg
Windows specifics:	Valid values for *library-specification*
See:	"SASHELP= System Option" in *SAS System Options: Reference*

Syntax

-SASHELP ("*library-specification-1*"...<"*library-specification-n*")>

Syntax Description

"*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.

You can use the APPEND and INSERT system options to add additional file specifications.

See Also

- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

SASINITIALFOLDER System Option: Windows

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.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
Windows specifics:	all

Syntax

-SASINITIALFOLDER *newfolder*

Required Argument

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, SAS determines the current folder by default. SAS uses the Sasuser folder as the default folders for the **Open and Save As** dialog boxes.

The current working folder is set according to information in "Determining the Current Folder When SAS Starts" on page 24.

SASUSER System Option: Windows

Specifies the name of the Sasuser library.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	c:\Users*userid*\Documents\My SAS Files\9.4
Windows specifics:	Valid values for *library-specification*; syntax

Syntax

-SASUSER ("*library-specification-1*"...<"*library-specification-n*">)

Syntax Description

"*library-specification-1*"... "*library-specification-n*"
specifies one or more valid Windows pathnames or environment variables that are associated with pathnames for a SAS 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 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 library is set to be equal to the Work library, which is temporary.

See Also

- "Profile Catalog " on page 36
- "Using the Sasuser Data Library" on page 169

SCROLLBARFLASH System Option: Windows

Specifies whether to allow the mouse or keyboard to focus on a scroll bar.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	NOSCROLLBARFLASH
Windows specifics:	all

Syntax

-SCROLLBARFLASH | **-NOSCROLLBARFLASH**

SCROLLBARFLASH | **NOSCROLLBARFLASH**

Syntax Description

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 can 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 84

SET System Option: Windows

Defines a SAS environment variable.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
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*">)

Syntax Description

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 contains spaces, enclose *value* in quotation marks.

Details

This action 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 only can 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 is 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.

You can use the APPEND and INSERT system options to add additional file specifications.

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 163
- "Using Environment Variables" on page 189
- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

SGIO System Option: Windows

Activates the Scatter/Gather I/O feature.

Valid in:	configuration file, SAS invocation
Category:	Files: SAS Files
PROC OPTIONS GROUP=	SASFILES
Default:	NOSGIO
Windows specifics:	all

Syntax

-SGIO | **-NOSGIO**

Syntax Description

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 the following SAS I/O files:

- those that contain a 4K-multiple page size (for example, 4096 or 8192) on 32–bit systems
- those that contain a 8K-multiple page size (for example, 8192 or 16384) on 64-bit systems
- those that were not created by using Version 6 of SAS
- those that 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. You can access more information about SGIO in the SGIO Usage document.

See Also

- "BUFNO System Option: Windows" on page 599
- "SAS Features That Optimize Performance" on page 252

SLEEPWINDOW System Option: Windows

Enables or disables the SLEEP window.

Valid in: configuration file, SAS invocation

Category: Environment Control: Display

PROC OPTIONS GROUP= ENVDISPLAY

Default: SLEEPWINDOW

Windows specifics: all

Syntax

-SLEEPWINDOW | **-NOSLEEPWINDOW**

Syntax Description

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: Windows” on page 491
- “WAKEUP Function: Windows” on page 493

SORTCUT System Option: Windows

Specifies the data size in number of observations above which SAS uses the host sort instead of the internal SAS sort.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Sort: Procedure Options

PROC OPTIONS GROUP= SORT

Default: 0

Restriction: Beginning in SAS 9.4M8 and in the October 2024 update for SAS Viya 3.5, this option is not used because the syncsort utility is not supported. If you specify this option, SAS writes a note to the SAS log that the SAS sort is used instead.

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

Note: You can also use the KB, MB, and GB syntax notation.

Required Arguments

***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: Windows" on page 687
- "SORTPGM System Option: Windows" on page 691

SORTCUTP System Option: Windows

Specifies the data size in bytes above which SAS uses the host sort instead of the internal SAS sort.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Sort: Procedure Options
PROC OPTIONS GROUP=	SORT
Default:	0
Restriction:	Beginning in SAS 9.4M8 and in the October 2024 update for SAS Viya 3.5, this option is not used because the syncsort utility is not supported. If you specify this option, SAS writes a note to the SAS log that the SAS sort is used instead.
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

Note: You can also use the KB, MB, and GB syntax notation.

Required Arguments

***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: Windows" on page 691
- "SORTCUT System Option: Windows" on page 685

SORTDEV System Option: Windows

Specifies the pathname to temporary files that are created by the host sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Sort: Procedure Options
PROC OPTIONS GROUP=	SORT
Default:	same location as -WORK, which is set in the configuration file
Restriction:	Beginning in SAS 9.4M8 and in the October 2024 update for SAS Viya 3.5, this option is not used because the syncsort utility is not supported. If you specify this option, SAS writes a note to the SAS log that the SAS sort is used instead.
Windows specifics:	all

Syntax

-SORTDEV *"pathname"*

SORTDEV =*"pathname"*

Required Argument

"*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: Windows" on page 717

SORTNAME System Option: Windows

Specifies the name of the host sort utility.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable, OPTIONS statement
PROC OPTIONS GROUP=	SORT
Default:	none
Restriction:	Beginning in SAS 9.4M8 and in the October 2024 update for SAS Viya 3.5, this option is not used because the syncsort utility is not supported. If you specify this option, SAS writes a note to the SAS log that the SAS sort is used instead.
Windows specifics:	all

Syntax

SORTNAME=*"host-sort-utility-name"*

-SORTNAME *host-sort-utility-name*

Details

The SORTNAME= option specifies the name of the default host sort utility.

See Also

"SORTPGM System Option: Windows" on page 691

SORTPARM System Option: Windows

Specifies the parameters for the host sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Sort: Procedure Options
PROC OPTIONS GROUP=	SORT
Default:	none
Restriction:	Beginning in SAS 9.4M8 and in the October 2024 update for SAS Viya 3.5, this option is not used because the syncsort utility is not supported. If you specify this option, SAS writes a note to the SAS log that the SAS sort is used instead.
Windows specifics:	all

Syntax

SORTPARM=*"SyncSort-parameters"*

-SORTPARM *"SyncSort-parameters"*

Required Argument

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: Windows

Specifies whether to use the SAS sort utility or to let SAS choose the sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Sort: Procedure Options
PROC OPTIONS GROUP=	SORT
Default:	BEST
Restriction:	Beginning in SAS 9.4M8 and in the October 2024 update for SAS Viya 3.5, this option is not used because the syncsort utility is not supported. If you specify this option, SAS writes a note to the SAS log that the SAS sort is used instead.
Windows specifics:	all

Syntax

-SORTPGM SAS | BEST

SORTPGM=SAS | BEST

Required Arguments

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, or the SAS sort. 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.

See Also

"SORT Procedure Statement: Windows" on page 529

SORTSIZE System Option: Windows

Specifies the amount of memory that is available to the SORT procedure.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Categories:	Sort: Procedure Options System Administration: Memory
PROC OPTIONS GROUP=	MEMORY SORT
Default:	1G
Windows specifics:	Default value
See:	"SORTSIZE= System Option" in *SAS System Options: Reference*

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

Required Arguments

***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 SAS work directory in which to store the data. The SORT procedure's algorithm can swap unneeded data more efficiently than Windows can.

If you can place the SAS data set that you want to sort in physical memory on your machine, then a sort in SAS is very efficient. Set SORTSIZE to be larger than the size of the data set. If you cannot fit the data set in physical memory, then set SORTSIZE to 1G or less. In addition, SORTSIZE should always be set to a value that is at least 8M smaller than MEMSIZE.

Comparisons

The SORTSIZE option is similar to the REALMEMSIZE option. SORTSIZE affects only the SORT procedure. REALMEMSIZE affects multiple procedures.

See Also

- "SORT Procedure Statement: Windows" on page 529
- "Improving Performance of the SORT Procedure" on page 254

SPLASH System Option: Windows

Specifies whether to display the splash screen (logo screen) when SAS starts.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	SPLASH
Windows specifics:	all

Syntax

-SPLASH | **-NOSPLASH**

-SPLASH ON | **-SPLASH OFF**

Syntax Description

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: Windows" on page 694

SPLASHLOC System Option: Windows

Specifies the location of the splash screen bitmap that appears when SAS starts.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	none
Windows specifics:	all

Syntax

-SPLASHLOC *DLL-name* <*res-number*> | *BMP-filename*

Syntax Description

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 *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.

STIMEFMT System Option: Windows

Specifies the format that is used to display the time on FULLSTIMER and STIMER output.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	M
Windows specifics:	all

Syntax

-STIMEFMT *value(s)*

STIMEFMT=*value(s)*

Required Arguments

value
specifies the options to use with STIMEFMT. The following options are available:

Timestamp options
The timestamp options are described below:

TS
specifies to always display the timestamp as part of STIMER and FULLSTIMER.

TSFULL
specifies to display the timestamp as part of FULLSTIMER. TSFULL is the default.

TSOFF turns off the timestamp for STIMER and FULLSTIMER.

Memory
is normally displayed as part of FULLSTIMER. The default memory output is displayed in KB. The following options for memory are available:

MEMFULL writes memory statistics as part of FULLSTIMER but not as part of STIMER.

MEM writes memory statistics as part of FULLSTIMER and STIMER.

KB writes memory in kilobytes.

MB writes memory in megabytes.

GB writes memory in gigabytes.

C adds commas to the numbers in the memory display.

NC does not add commas to the numbers in the memory display.

H, HOURS
specifies that SAS software display the STIMER output as `hours:minutes:seconds`.

Elapsed and CPU time
can be configured to display hours, minutes, seconds, or best fit in STIMER and FULLSTIMER.

Z | H | HOURS writes the time as hours:minutes:seconds.

M | MINUTES writes the time as minutes:seconds.

S | SECONDS writes the time as seconds.

HMS writes the format without leading zeros for hours and minutes.

Counters
specifies that additional counters can be displayed as part of FULLSTIMER.

E | ENABLE enables extra counters.

D | DISABLE disables extra counters.

Help
provides two values that are used to access help for the STIMEFMT option:

FMT lists the available timestamp formats.

OPT lists other option values that are available.

Details

STIMEFMT Basics

The STIMEFMT system enables you to customize the format of output produced by the STIMER and FULLSTIMER system options. You can perform the following tasks using STIMEFMT:

- list the formats that are available:

```
options stimefmt = fmt;
```

- list other options that are available:

```
options stimefmt = opt;
```

- turn the timestamp on or off for STIMER:

```
options stimefmt = tson | tsoff | tsfull;
```

- combine options as needed:

```
options stimefmt = (tson YYNNDDS);
```

- separate a memory value with commas:

```
options stimefmt = c;
```

- do not use commas when specifying values:

```
options stimefmt = nc;
```

- select a unit for memory:

```
options stimefmt = GB | MB | KB;
```

- turn on memory reporting for STIMER and FULLSTIMER:

```
options stimefmt = mem;
```

- set the time display in the timestamp:

```
options stimefmt = TOD | TIME | TIMEAMPM;
(TOD and TIME specify military time.)
```

- control the display of CPU or real time by using HOURS or MINUTES

Formats for Displaying the Timestamp

The format of timestamp can be set to standard formats that are supported by SAS. These formats include the following:

```
ABS.           (Absolute seconds since Jan. 1, 1970)

DATE.          DATE9.

DDMMYY.         DDMMYY10.     DDMMYYB.
DDMMYYB10.      DDMMYYC.      DDMMYYC10.
DDMMYYD.        DDMMYYD10.    DDMMYYN.
DDMMYYN10.      DDMMYYP.      DDMMYYP10.
DDMMYYS.        DDMMYYS10.
```

```
ISO.            (ISO Standard Time)

MMDDYY.          MMDDYY10.     MMDDYY.
MMDDYYB10.       MMDDYYC.      MMDDYYC10.
MMDDYYD.         MMDDYYD10.    MMDDYYN.
MMDDYYN8.        MMDDYYP.      MMDDYYP10.
MMDDYYS.         MMDDYYS10.

NLDATM.          NLDATMAP.

YYMMDD.           YYMMDD10.     YYMMDDB.
YYMMDDB10.        YYMMDDC.      YYMMDDC10.
YYMMDDD.          YYMMDDD10.    YYMMDDN.
YYMMDDN8.         YYMMDDP.      YYMMDDP10.
YYMMDDS.          YYMMDDS10.

TOD.               (Writes time as military time.)
TIME.              (Writes time as military time.)
TIMEAMPM.          (Writes time as AM and PM.)
```

The syntax for the OPTIONS statement is listed below:

```
options stimefmt = fmt;
```

where *fmt* is a valid SAS format.

Using Multiple Values for the STIMEFMT Option

The STIMEFMT option can specify multiple values at the same time to enable you to set multiple settings. Multiple values must be enclosed in parentheses, as the following example shows:

```
options stimefmt = (h YYMMDD. gb c);
```

Displaying the Settings for the STIMEFMT Option

PROC OPTIONS always displays the current state of all settings for STIMEFMT. The following example shows log output when you execute PROC OPTIONS:

```
proc options option=stimefmt;
run;
```

Example Code 21.1 *Log Output from PROC OPTIONS*

```
SAS (r) Proprietary Software Release 9.3  TS1B0

 STIMEFMT=(NLDATM2. HMS TIMEAMPM KB MEMFULL TSFULL NC)
                  Specified the output format for FULLSTIMER and STIMER.
                  This controls the timestamp, memory, CPU and elapsed time.
```

Resetting STIMEFMT to the Default Values

You can reset the setting for STIMEFMT to the default values by executing the following OPTIONS statement:

```
options stimefmt = normal;
```

See Also

"The SAS Log" in *SAS Programmer's Guide: Essentials*

STIMER System Option: Windows

Writes a subset of system performance statistics to the SAS log.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	STIMER
Windows specifics:	Reported statistics

Syntax

-STIMER | **-NOSTIMER**

STIMER | **NOSTIMER**

Syntax Description

STIMER
writes real time and CPU time to the SAS log.

NOSTIMER
does not write real time and CPU time to the SAS log.

Details

The STIMER system option is printed to the SAS log the amount of time it took for SAS to complete a DATA step or procedure task.

Here is an example of STIMER output:

```
real time       0.96 seconds
cpu time        0.01 seconds
```

STIMER displays the following statistics:

Table 21.7 *Description of STIMER Statistics*

Statistic	Description
real time	the amount of time spent to process the SAS job. Real time is also referred to as elapsed time.
CPU Time	the total time spent to execute your SAS code and spent to perform system overhead tasks on behalf of the SAS process. This value is the combination of the user CPU and system CPU statistics from FULLSTIMER.

Note: Starting in SAS 9, some procedures use multiple threads. On computers with multiple CPUs, the operating system can run more than one thread simultaneously. Consequently, CPU time might exceed real time in your STIMER output.

For example, a SAS procedure could use two threads that run on two separate CPUs simultaneously. The value of CPU time would be calculated as the following:

```
CPU1 time + CPU2 time = total CPU time
1 second + 1 second = 2 seconds
```

Since CPU1 can run a thread at the same time that CPU2 runs a separate thread for the same SAS process, you can theoretically consume 2 CPU seconds in 1 second of real time.

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: Windows” on page 621
- “Optimizing System Performance” in *SAS Programmer’s Guide: Essentials*

SYSGUIFONT System Option: Windows

Specifies a font to use for the button text and the descriptive text.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	depends upon display settings
Windows specifics:	all

Syntax

-SYSGUIFONT "*font-name*" <*font-size*>

Syntax Description

"*font-name*"
specifies the name of the font for text in screen and dialog box text elements. This name 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, and 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: Windows" on page 616
- "Selecting Fonts" on page 83

SYSIN System Option: Windows

Specifies a batch mode source file.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
Windows specifics:	Valid values for *file-specification*

Syntax

-SYSIN *file-specification* | **-NOSYSIN**

Syntax Description

SYSIN *file-specification*
specifies to start SAS and submit the file in batch mode. The value of *file-specification* must be a valid Windows file name.

NOSYSIN
specifies to start SAS in batch mode, but do not submit any files. This option 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: Windows

Specifies a character string that can be passed to SAS programs.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
Windows specifics:	Valid values and syntax for *characters*

See: *SAS Macro Language: Reference*

Syntax

-SYSPARM *characters*

SYSPARM= *characters*

Syntax Description

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: Windows

Specifies a destination printer for printing SAS output.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Log and Procedure Output Control: Procedure Output
PROC OPTIONS GROUP=	LISTCONTROL
Default:	Default system printer
Windows specifics:	all

Syntax

-SYSPRINT *"printer-name"*<*"destination"*>

SYSPRINT=*"printer-name"*<*"destination"*>

Syntax Description

"*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*"
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 can 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 dialog boxes 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 208
- "PRTPERSISTDEFAULT System Option: Windows" on page 667
- "SYSPRINTFONT System Option: Windows" on page 705

SYSPRINTFONT System Option: Windows

Specifies the font to use when SAS is printing to the current default 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
Default:	none
Windows specifics:	all
See:	"SYSPRINTFONT= System Option" in *SAS System Options: Reference*

Syntax

-SYSPRINTFONT ("*font-name*" <BOLD | NORMAL> <REGULAR | ITALIC> <*character-set*
> <*point-size*> <<NAMED*printer-name*""" | UPRINT="*printer-name*" | DEFAULT | ALL>>)

SYSPRINTFONT="*font-name*" <BOLD | NORMAL> <REGULAR | ITALIC> <*character-set*
> <*point-size*> <NAMED*printer-name*""" | UPRINT="*printer-name*" | DEFAULT | ALL>

Syntax Description

"font-name"
specifies the name of the font to use for printing. This name 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, and 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 value 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.

UPRINT="*printer-name*"
specifies a Universal Printer to which these settings apply. UPRINT is valid only for printers that are listed in the SAS registry. The *printer-name* must exactly match the name shown in the **Print Setup** dialog box, except that the printer name is not case sensitive. If the Universal Prints is more than one word, *printer-name* must be enclosed in single or double quotation marks. The quotation marks are stored with the printer-name.

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 System Options window. Parentheses 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 is

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 does not specify a valid font for the printer, and the current display font is scalable, then SAS uses the display font to be printed.
4. If the current display font is not scalable, SAS uses 10-point SAS Monospace.

5 If the SAS Monospace font is not available, SAS uses the printer's default font to be printed.

Note: To ensure that row and column separators and boxed tabular reports are printed legibly when using the standard forms characters, you must use these resources:

- the SAS Monospace or the SAS Monospace Bold font
- a printer that supports TrueType fonts

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:")
```

Example 3: Specifying a Font to a Universal Printer on the SAS Command Line

This example specifies the Albany AMT font for the PDF Universal Printer:

```
sysprintfont=('courier' 11 uprint='PDF')
```

See Also

"SYSPRINT System Option: Windows" on page 703

TOOLDEF System Option: Windows

Specifies the Toolbox display location.

Valid in: configuration file, SAS invocation

Category: Environment Control: Display

PROC OPTIONS GROUP= ENVDISPLAY

Default: TOP RIGHT

Windows specifics: all

Syntax

-TOOLDEF TOP | CENTER | BOTTOM <LEFT | CENTER | RIGHT>

Syntax Description

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 toolbar instead of the Toolbox.

See Also

- "Customizing a Toolbar" on page 96
- "Using the Toolbar to Issue Commands" on page 63

TOOLSMENU System Option

Specifies whether the Tools menu is included in SAS windows.

Valid in: Configuration file, SAS invocation

Category: Environment Control: Display

PROC OPTIONS GROUP= ENVDISPLAY

Syntax

TOOLSMENU | **NOTOOLSMENU**

Syntax Description

TOOLSMENU
specifies that the Tools menu is included in SAS windows.

NOTOOLSMENU
specifies that the Tools menu is not included in SAS windows.

UNIVERSALPRINT System Option

Specifies whether to enable menus for Universal Printing and to set up printing defaults.

Valid in: Configuration file, SAS invocation

Category: Log and Procedure Output Control: ODS Printing

PROC OPTIONS GROUP= ODSPRINT

Syntax

UNIVERSALPRINT | **NOUNIVERSALPRINT**

Syntax Description

UNIVERSALPRINT
routes all printing through the Universal Print services.

Alias UPRINT

NOUNIVERSALPRINT
disables printing through the Universal Print services.

Alias NOUPRINT

Details

Universal Printing services provides interactive and batch printing capabilities to SAS applications and procedures. The ODS PRINTER destination uses Universal Print services as needed, whether the UNIVERSALPRINT option or the NOUNIVERSALPRINT option is set.

When UNIVERSALPRINT is specified, SAS sets the following environment:

- The Universal Printing menu items and dialog boxes are enabled in the windowing environment.
- The PRINTERPATH= system option is set to a default, non-null value, usually PostScript Level 1, when SAS starts.
- The SYSPRINT= system option is ignored. The value of the PRINTERPATH= system option specifies the output destination.
- Setting PRINTERPATH="" sets the value of PRINTERPATH to the default value.

To use the Windows printing environment in SAS, set NOUNIVERSALPRINT. SAS sets the following environment:

- The initial value of the PRINTERPATH= system option is null and SAS enables the Windows print menus and dialog boxes.
- If the PRINTERPATH= system option is set to a non-null value, SAS uses Universal Printing. SAS uses the Windows print dialog boxes but the referenced printer in the dialog box is not used.
- If PRINTERPATH="", SAS uses the Windows print dialog boxes and the SYSPRINT= system option specifies the output destination.

See Also

- "Universal Printing" in *SAS Programmer's Guide: Essentials*

System Options:

- "PRINTERPATH= System Option" in *SAS System Options: Reference*
- "SYSPRINT System Option: Windows" on page 703
- "UPRINTCOMPRESSION System Option" in *SAS System Options: Reference*

UPRINTMENUSWITCH System Option: Windows

Enables the universal print commands on the File menu.

Valid in: configuration file, SAS invocation

Category: Log and Procedure Output Control: ODS Printing

PROC OPTIONS GROUP= ODSPRINT

Default: NOUPRINTMENUSWITCH

Windows specifics: all

Syntax

-UPRINTMENUSWITCH | **-NOUPRINTMENUSWITCH**

Syntax Description

UPRINTMENUSWITCH
specifies that the print commands on the File menu invoke the **Universal Printing** dialog boxes.

NOUPRINTMENUSWITCH
specifies that the print commands on the File menu invokes 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 does not invoke the Universal Printing menus and dialog boxes.

See Also

"Introduction to Printing in SAS within the Windows Environment" on page 208

USER System Option: Windows

Specifies the name of the default permanent SAS library.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment Control: Files

PROC OPTIONS GROUP= ENVFILES

Default: none

Windows specifics: Valid values for *library-specification*

See: "USER= System Option" in *SAS System Options: Reference*

Syntax

-USER "*library-specification*"

USER="*library-specification*"

Required Argument

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 are permanently stored in the specified library. If you want to create a temporary data set, use a two-level name for the data set, and the first part being Work (for example, `work.tempdata`).

USERICON System Option: Windows

Specifies the pathname of the resource file associated with your user-defined icon.

Valid in: configuration file, SAS invocation

Category: Environment Control: Display

PROC OPTIONS GROUP= ENVDISPLAY

Default: none

Windows specifics: all

Syntax

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

Required Arguments

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: Windows

Controls whether SAS writes the settings of all the system options specified in the configuration file to either the terminal or the batch log.

Valid in:	configuration file, SAS invocation
Category:	Log and Procedure Output Control: SAS Log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOVERBOSE
Windows specifics:	Amount of information reported

Syntax

-VERBOSE | **-NOVERBOSE**

Syntax Description

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. NOVERBOSE 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 whether you have an error in your system option specifications.

See Also

"The SAS Log" in *SAS Programmer's Guide: Essentials*

VIEWMENU System Option

Specifies whether the View menu is included in SAS windows.

Valid in:	Configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY

Syntax

VIEWMENU | **NOVIEWMENU**

Syntax Description

VIEWMENU
specifies that the View menu is included in SAS windows.

NOVIEWMENU
specifies that the View menu is not included in SAS windows.

WEBUI System Option: Windows

Specifies to enable web enhancements.

Valid in:	configuration file, SAS invocation
Category:	Input Control: Data Processing
PROC OPTIONS GROUP=	INPUTCONTROL
Default:	NOWEBUI
Windows specifics:	all

Syntax

-WEBUI | **-NOWEBUI**

Syntax Description

WEBUI
specifies to enable web enhancements.

NOWEBUI
specifies to disable web enhancements.

Details

If you have installed Microsoft Internet Explorer and specify the WEBUI system option, certain windows, such as the 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: Windows

Specifies to include or suppress the Window menu in windows that display menus.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	NOWINDOWSMENU
Windows specifics:	all

Syntax

-WINDOWSMENU | **-NOWINDOWSMENU**

WINDOWSMENU | **NOWINDOWSMENU**

Syntax Description

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: Windows" on page 597

WORK System Option: Windows

Specifies the location of the Work library.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	!TEMP\SAS Temporary Files
Windows specifics:	all
See:	"WORK= System Option" in *SAS System Options: Reference*

Syntax

-WORK *filename* | *directory*

Required Arguments

filename
specifies a file that contains a list of directories and optional keywords. SAS chooses a directory from the list in the file. That directory is the location for the Work library for the current SAS session.

directory
specifies a directory as the location for the Work library for the current SAS session.

Details

The Basics

When you use the file name option, SAS selects a directory to use as the location for the Work library. SAS randomly selects a directory or selects a directory based on available space. You use the METHOD keyword to make your selection.

When you use the directory option, SAS continues its initialization, using the specified directory as the location for the Work library.

Making the Allocation of Work Libraries Dynamic

The file name option contains a list of directories that are used for the Work library. Individual SAS Work libraries reside in a single directory. You use

METHOD=RANDOM to specify that the directory for the Work library is randomly chosen from the list of directories. SAS selects one directory per session as the location for the Work library. This selection enables you to balance the I/O load across multiple hardware systems. You use METHOD=SPACE to specify the directory that has the most available space. When the METHOD keyword is not specified, SAS defaults to randomly selecting a directory.

Examples

Example 1: Spreading a Processing Load across Multiple Volumes of Different Disks

The configuration file (Default location: **C:\Program Files\SASHome \SASFoundation\9.4\nls\en\sasv9.cfg**) or command line includes the following:

-WORK "C:\SASWork\created file name.txt"

The following example shows how to spread an I/O processing load across multiple volumes of different disks. In this case, you use METHOD=RANDOM. A text file named **C:\SASWork\created file name.txt** contains the following information:

```
W:\disk1\sastempfiles
W:\disk2\sastempfiles
W:\disk3\sastempfiles
Method=Random
```

Note: The Work library for a SAS session is placed on either disk1, disk2, or disk3 at random.

Example 2: Choosing the Directory That Has the Most Free Space

When you process your data, you can choose the directory that has the most free space. In this case, you use METHOD=SPACE. In this example, **C:\SASWork \created file name.txt** contains the following directories:

```
W:\disk1\sastempfiles
W:\disk2\sastempfiles
W:\disk3\sastempfiles
Method=Space
```

Note: The Work library is placed on the disk with the most free space.

See Also

"Work Data Library " on page 38

XCMD System Option: Windows

Specifies that the X command is valid in the current SAS session.

Valid in:	configuration file, SAS invocation
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Defaults:	SAS 9.4 in SAS Windowing Environment: XCMD SAS 9.4 in SAS Studio: NOXCMD SAS Viya: NOXCMD
Windows specifics:	all

Syntax

-XCMD | **-NOXCMD**

-XCMD ON | **-XCMD OFF**

Syntax Description

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.

If you specify NOXCMD, the following are disabled:

- PIPE and NAMEPIPE device types in the FILENAME statement
- CALL SYSTEM routine
- X command
- Dynamic Data Exchange (DDE)

- %SYSEXEC macro
- SYSTASK statement
- PIPE and NAMEPIPE device types in the FILENAME function.

See Also

"X Command: Windows" on page 444

XMIN System Option: Windows

Specifies to open the application specified in the X command in a minimized state or in the default active state.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	NOXMIN
Windows specifics:	all

Syntax

-XMIN | **-NOXMIN**

XMIN | **NOXMIN**

Syntax Description

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 enables you to open an application specified in the X command in a minimized state or in the default active state.

XSYNC System Option: Windows

Controls whether an X command or statement executes synchronously or asynchronously.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	XSYNC
Windows specifics:	all

Syntax

-XSYNC | **-NOXSYNC**

XSYNC | **NOXSYNC**

Syntax Description

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. XSYNC 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 45
- “XWAIT System Option: Windows” on page 722
- “X Statement: Windows” on page 565
- “X Command: Windows” on page 444
- “CALL SYSTEM Routine: Windows” on page 464
- “Macro Statements” on page 733

XWAIT System Option: Windows

Specifies whether you have to type EXIT at the DOS prompt before the DOS shell closes.

Valid in:	configuration file, SAS invocation, OPTIONS statement
Category:	Environment Control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	XWAIT
Windows specifics:	all

Syntax

-XWAIT | **-NOXWAIT**

XWAIT | **NOXWAIT**

Syntax Description

XWAIT
specifies that you must type EXIT to return to your SAS session. XWAIT 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 applies only 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 45
- "XSYNC System Option: Windows" on page 721
- "X Statement: Windows" on page 565
- "X Command: Windows" on page 444
- "CALL SYSTEM Routine: Windows" on page 464
- "Macro Statements" on page 733

22

Length and Precision of Variables under Windows

Length and Precision of Variables under Windows 725
Numeric Variables 725
Character Variables 727

Length and Precision of Variables under Windows

For detailed information about how SAS stores representations of numeric and character data, see *SAS Programmer's Guide: Essentials*. Data representation issues vary among different operating environments; the topics in this section discuss how data are represented in SAS under Windows.

- "Numeric Variables" on page 725
- "Character Variables" on page 727

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 accurate to approximately 15 digits. Depending on the number, the precision can be 16 digits of accuracy. For more information about the representation of the LONG REAL

Windows data type, see Intel developer website. Table 22.1 on page 726 specifies the significant digits and largest integer values that can be stored in SAS numeric variables.

Table 22.1 *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
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 variables 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 you are storing real numbers, not integers, you should specify a length of 8 bytes to ensure the most precision as possible. If the value of a variable becomes large or has many significant digits, you can 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 is limited by the first encountered limitation:

- the observation length
- the total storage possible for names, labels, and metadata
- the amount of available memory on the machine where the data set is stored.

You can define a data set with an observation length of up to 2GB on a 32-bit platform and approximately 2*46 bytes on a 64-bit platform. The observation length cannot exceed the value of the BUFSIZE option.

Assuming a single-byte character set, and assuming that you use the maximum 352 bytes that are possible for name, label, and other data for each variable, you can have a maximum of approximately 1,350,000 variables. If the names, labels, and format names are shorter, you can have more than 66,666,666. A maximum of 1GB is required to store all the variable names and other metadata (data set label, compression name, and other data). The 352-byte maximum is the result of adding 32 bytes each for formats, informats, and variable names to the 256 bytes for label values.

Assuming that the above limits are not exceeded the maximum possible number of variables is 2GB on 64-bit hosts.

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 is limited by the first encountered limitation:

- the observation length
- the total storage possible for names, labels, and metadata
- the amount of available memory on the machine where the data set is stored.

You can define a data set with an observation length of up to 2GB on a 32-bit platform and approximately 2*46 bytes on a 64-bit platform. The observation length cannot exceed the value of the BUFSIZE option.

Assuming a single-byte character set, and assuming that you use the maximum 352 bytes that are possible for name, label, and other data for each variable, you can have a maximum of about 4,050,000 variables. If the names, labels, and format names are shorter, you can have more than 200,000,000. There is aA maximum of 1GB is required to store all the variable names and other metadata (data set label, compression name, and other data). The 352-byte maximum is the result of adding 32 bytes each for formats, informats, and variable names to the 256 bytes for label values.

Assuming that the above limits are not exceeded, the maximum possible number of variables is 2GB on 64-bit hosts.

23

SAS Macro Facility under Windows

SAS Macro Facility under Windows 729
Automatic Macro Variables 730
Macro Statements 733
Macro Functions 734
Autocall Libraries 735
Overview of Autocall Libraries 735
SASAUTOS System Option 735

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 *SAS Macro Language: Reference*.

The following aspects of the macro facility have details that are specific to Windows:

- "Automatic Macro Variables" on page 730
- "Macro Statements" on page 733
- "Macro Functions" on page 734
- "Autocall Libraries" on page 735

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.

  Note: When ERRORCHECK=NORMAL, the return code is 0 even if an error exists in a LIBNAME or FILENAME statement, or in a LOCK statement in SAS/SHARE software. Also, the SAS job or session does not end when the %INCLUDE statement fails due to a nonexistent file. For more information, see the ERRORCHECK= system option in *SAS System Options: Reference*.

SYSDEVIC
: gives the name of the current graphics device. The current graphics device is determined by the DEVICE system option. Contact your on-site SAS support personnel to determine which graphics devices are available at your site. For information about the DEVICE system option, see "DEVICE System Option: Windows" on page 606 and *SAS System Options: Reference*.

SYSENV
: can contain the values `FORE` or `BACK` under Windows. If you are running EG on Windows, the value of SYSENV is `BACK`.

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.

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. You do not have to enter **`exit`** to return to your SAS session. The value of "This Windows status is" is 1, and the

value for “The corrected Windows status is” is 0. If you run this example with the XWAIT option, you would need to enter **`exit`** before SAS would run the code. After you enter **`exit`**, a value of 9009 is returned for the statement “This Windows status is”, and 0 is the value for “The corrected Windows status is”. 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. Here are some possible return values:

  W32_7HOME
  : Microsoft Windows 7 Home

  W32_7PRO
  : Microsoft Windows 7 Professional

  W32_8HOME
  : Microsoft Windows 8 Home

  W32_8PRO
  : Microsoft Windows 8 Professional

  W32_81HOME
  : Microsoft Windows 8.1 Home

  W32_81PRO
  : Microsoft Windows 8.1 Professional

  W32_10HOME
  : Microsoft Windows 10 Home

  W32_10PRO
  : Microsoft Windows 10 Professional

  W32_SRV08
  : Windows Server 2008 Standard

  W32_ESRV08
  : Windows Server 2008 Enterprise

  W32_DSRV08
  : Windows Server 2008 Datacenter

  W32_S08R2
  : Windows Server 2008 R2 Standard

  W32_ES08R2
  : Windows Server 2008 R2 Enterprise

  W32_DS08R2
  : Windows Server 2008 R2 Datacenter

  W32_FSRV12
  : Windows Server 2012 Foundation

  W32_ESRV12
  : Windows Server 2012 Essentials

W32_SRV12
Windows Server 2012 Standard

W32_DSRV12
Windows Server 2012 Datacenter

W32_FS12R2
Windows Server 2012 R2 Foundation

W32_ES12R2
Windows Server 2012 R2 Essentials

W32_SR12R2
Windows Server 2012 R2 Standard

W32_DS12R2
Windows Server 2012 R2 Datacenter

W32_SRV16
Windows Server 2016 Standard

W32_DSRV16
Windows Server 2016 Datacenter

W32_SRV19
Windows Server 2019 Standard

W32_DSRV19
Windows Server 2019 Datacenter

X64_7HOME
Microsoft Windows 7 Home

X64_7PRO
Microsoft Windows 7 Professional

X64_8HOME
Microsoft Windows 8 Home

X64_8PRO
Microsoft Windows 8 Professional

X64_81HOME
Microsoft Windows 8.1 Home

X64_81PRO
Microsoft Windows 8.1 Professional

X64_10HOME
Microsoft Windows 10 Home

X64_10PRO
Microsoft Windows 10 Professional

X64_SRV08
Windows Server 2008 Standard

X64_ESRV08
Windows Server 2008 Enterprise

X64_DSRV08
Windows Server 2008 Datacenter

X64_S08R2
Windows Server 2008 R2 Standard

X64_ES08R2
Windows Server 2008 R2 Enterprise

X64_DS08R2
Windows Server 2008 R2 Datacenter

X64_FSRV12
Windows Server 2012 Foundation

X64_ESRV12
Windows Server 2012 Essentials

X64_SRV12
Windows Server 2012 Standard

X64_DSRV12
Windows Server 2012 Datacenter

X64_FS12R2
Windows Server 2012 R2 Foundation

X64_ES12R2
Windows Server 2012 R2 Essentials

X64_SR12R2
Windows Server 2012 R2 Standard

X64_DS12R2
Windows Server 2012 R2 Datacenter

X64_SRV16
Windows Server 2016 Standard

X64_DSRV16
Windows Server 2016 Datacenter

X64_SRV19
Windows Server 2019 Standard

X64_DSRV19
Windows Server 2019 Datacenter

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 “Overview of Running Windows or MS-DOS Commands from within SAS” on page 45 . 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: Windows" on page 721 and "XWAIT System Option: Windows" on page 722.

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:\Windows\system32\cmd.exe
```

Autocall Libraries

Overview of 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, set the SAS system option MAUTOSOURCE. 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. 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: Windows" on page 675.

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 can also edit your SAS configuration file to add your autocall library to the library concatenation that is supplied by SAS, 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 in which 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

Appendixes

Appendix 1
Using Dynamic Data Exchange under Windows 739

Appendix 2
Error Messages for SAS under Windows 755

Appendix 3
Graphics Considerations under Windows 765

Appendix 4
Default Key Settings for Interactive SAS Sessions under Windows 767

Appendix 5
Using EBCDIC Data on ASCII Systems 779

Appendix 6
SCL Methods for Automating OLE Objects under Windows 795

Appendix 7
Cleanwork Utility 807

Appendix 8
SASIOTEST Utility 813

Appendix 1

Using Dynamic Data Exchange under Windows

Dynamic Data Exchange (DDE) – A Legacy Method ... **740**

Overview of DDE ... **742**

DDE Syntax within SAS ... **743**

Referencing the DDE External File ... **744**
- Overview of Referencing the DDE External File ... 744
- Using the DDE Triplet ... 744
- Controlling Another Application Using DDE ... 745

DDE Examples ... **746**
- Overview of DDE Examples ... 746
- Using the X Command to Open a DDE Server ... 746
- Using DDE to Write Data to Microsoft Excel ... 746
- Using DDE to Write Data to Microsoft Word ... 747
- Using DDE to Read Data from Microsoft Excel ... 747
- Using DDE to Read Data from Microsoft Word ... 748
- Using DDE and the SYSTEM Topic to Invoke Commands in an Application Using Excel ... 748
- Using the NOTAB Option with DDE ... 749
- Using the DDE HOTLINK ... 750
- Using the !DDE_FLUSH String to Transfer Data Dynamically ... 751
- Using Macro Variables to Issue DDE Commands ... 752
- Reading Missing Data ... 752

Dynamic Data Exchange (DDE) – A Legacy Method

DDE is a legacy method, and SAS does not encourage using DDE. Instead, consider the alternatives that are supported by newer SAS features:

- SAS on a remote server
- SAS Enterprise Guide
- local client on the PC

When you modernize to a centralized SAS environment, be aware of several challenges. For example, if your SAS session runs on a remote machine (often a machine that is not running on Windows), then SAS cannot use DDE to communicate to your local Microsoft Excel application. DDE works only between two Windows processes (SAS and Excel) on the same machine.

However, SAS programs that use DDE can still work, as long as you situate your SAS software and its DDE partner (usually Microsoft Excel) to run on the same Windows PC.

If you have processes that rely on DDE but need to move to a client/server SAS environment in a short amount of time, take the following approach. Maintain a designated machine with SAS for Windows to do your DDE basic work, even as you move other processes to an enterprise SAS environment.

DDE is a 16-bit Windows technology used for communication between applications. The DDE protocol relies upon sending and posting specific window messages between two windows that can be owned by the same application or by different applications.

In 32-bit and 64-bit Windows, there are more efficient and secure methods for inter-process communication. DDE support in Office applications can be disabled when it is not needed to prevent malware from using automation via DDE. DDE support can be enabled in Office applications to maintain compatibility with legacy applications that still rely upon DDE for Office automation.

Office provides a COM object model (OM) that can be used for automating Office applications. New applications should use the OM for Office automation rather than using DDE. Existing applications should be updated to use the Office OM for automation purposes. KB article 257757 is specific to automating Office applications from Windows service or a process started by a Windows service, where there is no interactive user. Running and automating Office applications from a service context is not considered a supported scenario.

DDE is often disabled by Microsoft on a system by default since it can be leveraged as a method to introduce malware and has been classified as a security risk. Because of the limitations of this aging technology and the potential security and

stability concerns regarding its use, SAS does not support using DDE on a Windows server or from within any multi-client environment.

Here are some Microsoft security articles:

- Microsoft Security Advisory 4053440: Securely opening Microsoft Office documents that contain Dynamic Data Exchange (DDE) fields
- Microsoft Office Defense in Depth Update: Security Advisory
- Considerations for server-side Automation of Office

DDE works between two Windows processes on the same machine, and the only supported use of this technology from SAS is from within an interactive desktop log in session on a workstation where SAS and Excel or Word are both executing under the user's account and are the only application processes currently executing. If either application is executing on a non-visible and non-interactive desktop, as might be the case when executing as a Windows service, the use of DDE is not supported. The two supported execution modes for SAS with regard to DDE are either from within a SAS DMS session or by classic batch operation from within a Windows command shell. Using DDE with batch mode that is accessed by the **Right-Click** pop-up menu that appears when selecting a .sas file from the Windows **File Explorer** is not supported.

Note: This information applies to terminal services and remote desktop.

Multi-client environments refer to terminal server environments, such as Windows server with remote desktop services enabled or Windows client machines when using the switch-user option. Each interactive user log on runs in its own terminal session. Each terminal session has its own instance of the window manager, initialized by CSRSS.EXE, which manages the data associated with Windows and other users created by processes running in that terminal session. The data managed by the window manager is accessible only by processes running in the associated terminal session.

Windows created by processes in one terminal session cannot be accessed by processes running in another terminal session. DDE is used for inter-process communication between processes running in the same terminal session. It cannot be used to communicate with a process running in a different terminal session.

There is no restriction on processes running in different sessions from simultaneously using DDE, as processes in each terminal session are using the instance of the window manager for that session. Consider the scenario where there is a client application, DDEClient.exe, and a server application, DDEServer.exe, that use DDE for inter-process communication. User A logs on, in terminal session #1, and starts an instance of the client and server applications. User B then logs on, in terminal session #2, and also starts an instance of the client and server application. The client application running in session #1 uses DDE to communicate with the server application running in session #1, not the server application running in session #2. The window objects created by the client and server application running in session #1 are managed by the instance of the window manager associated with that session. Processes running in session #1 have no knowledge of the windows objects created by instance of the application running in session #2.

Here are some alternatives to DDE:

SAS Add-In for Microsoft Office
: Use Excel to pull your SAS content into the spreadsheet. You can use SAS stored processes to encapsulate whatever SAS operation you need.

Office Scripting
: You can use Office scripting to automate the process like a batch job, so that the process can run unattended. Those who use your Excel documents do not need to install the SAS Add-In for Microsoft Office in order to view the results.

PROC EXPORT
: supports adding sheets to existing files or replacing entire sheets.

Scripts
: enables the use of scripts like VBS and PowerShell to plug SAS data results into a spreadsheet as a post-process, which users can spawn within their SAS programs.

ODS EXCEL
: For batch SAS users, ODS tagsets. ExcelXP and ODS EXCEL can be used to place formatted report content into an Excel spreadsheet. See ODS EXCEL for an example.

Overview of 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 use is to acquire data from a Windows spreadsheet or database application.

You can access applications that are supported by Dynamic Data Exchange (DDE) by accessing the following URL. Word and Excel applications are supported. The PowerPoint application is not supported.

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 286 .

DDE Syntax within SAS

To use DDE in SAS, issue a FILENAME statement with the following syntax:

FILENAME *fileref* DDE '*DDE-triplet*' <DDE-options>;

fileref
is a valid fileref (as described in "Referencing External Files" on page 186).

DDE
is the device-type keyword that tells SAS that 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 745 .

HOTLINK
instructs SAS to use the DDE HOTLINK. For an example of using this option, see "Using the DDE HOTLINK" on page 750 .

LRECL=*record-length*
specifies the record length (in bytes). Under Windows, the default is 32767. The value of *record-length* can range from 1 to 1,073,741,823 (1 gigabyte).

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 749 .

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. If LRECL is not specified, by default SAS reads 32767 bytes at a time from the file.
P	indicates print format.
V|D	indicates variable format. This is the default.

CAUTION
Use caution when using DDE with data values that are blank or missing. For sample code, see "Reading Missing Data" on page 752 .

Referencing the DDE External File

Overview of 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 that you run. For information about 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 that 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 application 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`.

Suppose you want to use SAS to populate the first four rows and two 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 that 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 297 .

DDE Examples

Overview of 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:\program files\microsoft office\office14\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 send the data, 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 Word, you would generally use Visual Basic commands such as FileOpen.Name, FileSave, FileClose, and Insert. If the PUT statement contains a macro that Word does not understand, you 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, SHEET1, is loaded, and data is 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:\program files\microsoft office\office14\excel.exe"';

/* Sleep for 60 seconds to give */
/* Excel time to start 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 can also 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 writes 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, entering 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. The values currently stored in the spreadsheet cells when 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 how DDE is used, including the capacity to transfer data dynamically through the DATA step. The following example creates a Macro Sheet in Microsoft Excel. Commands are then written to the Macro Sheet, which is renamed Sheet1 to NewSheet. After writing these commands, through the use of !DDE_FLUSH , the Excel Macro can be executed in the same DATA Step as it is written.

```
filename cmds dde 'excel|system';
data _null_;
file cmds;
/* Insert an Excel Macro Sheet */
put '[workbook.insert(3)]';
run;

/* Direct the Output to the Newly created Macro Sheet */
filename xlmacro dde 'excel|macro1!r1c1:r5c1' notab;

data _null_;
file xlmacro;
put '=workbook.name("sheet1","NewSheet")';
put '=halt(true)';
/* Dump the contents of the buffer, allowing us to both write and */
/* execute the macro in the same DATA Step                        */
```

```
put '!dde_flush';
file cmds;
/* Run Macro1 */
put '[run("macro1!r1c1")]';
put '[error(false)]';
/* delete the Macro Sheet */
put '[workbook.delete("macro1")]';
run;
```

Using Macro Variables to Issue DDE Commands

This example illustrates the use of a Macro Variable to issue a command to Microsoft Excel. In the example, the Macro Variable, excelOne, is being used in place of the Excel Workbook location **C:\test.xls**. Since macro triggers such as ampersands and percents are treated as text within single quotation marks, a Macro quoting function must be used. %STR is used to mask each individual apostrophe separately. Anytime you have an unmatched apostrophe or parenthesis then it must be preceded by a percent sign and since each apostrophe needs to be treated independently of each other the percents are needed. Once %STR has hidden the apostrophe, the macro variable &excelOne resolves. %UNQUOTE is then used to remove what %STR has done and restores each apostrophe around the following resolved value.

```
'[open("C:\test.xls")]'
```

```
options mprint symbolgen;
filename cmds dde 'excel|system';

%let excelOne=C:\test.xls;

data _null_;
file cmds;
put %unquote(%str(%'[open("&excelOne")]%'));
run;
```

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 can 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 is 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 System Options: Reference*. 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 or DATALINES statement, whether your data were blank- or comma-delimited.

Appendix 2

Error Messages for SAS under Windows

Overview of SAS Error Messages 755
Return Codes and Completion Status 756
Accessing Files 757
Using SAS Features 758
Using OLE 759
Using Networks 760
Resolving Internal Errors 761
Resolving Operating System and Windows Error Messages 762
Initialization and Termination Error Messages 762

Overview of SAS Error Messages

This section contains completion codes and error messages that you might 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 about the ERRORLEVEL variable. The following table summarizes the values of the ERRORLEVEL variable.

Note: To check the ERRORLEVEL variable, start SAS using the START/ WAIT command. From a DOS window enter:

```
start/wait "title" "C:\Program Files\SASHome\SASFoundation\9.4\sas.exe"
```

Table A11.1 *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 that 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 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 761.

`Error: Date/time is in the future.`
The date or time stamp of the file that 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 that 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) might be unavailable. The requested file might not be available 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 that 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 762 .

`ERROR: Physical file does not exist filename.`
The file that 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 storage device.

`ERROR: Write access to member member-name is denied.`
You are trying to update a file on a write-protected storage device or you are trying to update a file marked as read-only.

Using SAS Features

This section describes errors that you can receive while using features of the SAS language and procedures under Windows. Always check the syntax of the statement or procedure that you are using. Also, if you do not get the results that 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 can 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 be set only in the SAS configuration file or in the SAS command. For more information, see "SAS Options under Windows" on page 572 .

`ERROR: Unknown configuration option option-name.`
Check your SAS command and SAS configuration file for an invalid SAS system option. For more information about system options under Windows, see "SAS Options under Windows" on page 572 .

`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 that you have attempted to load has no tools defined. At least one tool must be 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 that you can 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 might 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 box?`
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 cannot 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 can 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 that 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 cannot 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 that you can 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 can 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. If you receive these error messages, you might 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. You can also 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 ensure 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: Windows” on page 680 . Also see “Profile Catalog ” on page 36 .

`Work library is undefined. Sasuser library is undefined. Sashelp library is undefined.`
Check your SAS command or SAS configuration file to ensure 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 ensure 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 ensure 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 ensure that you have specified the directory path correctly for all SAS libraries. Also check to determine whether 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 corrupted or have been inadvertently deleted.

`601 Invalid setinit information.`
Check your setinit information for errors. For more information, see the installation instructions for SAS 9.4 for Windows.

`602 CORE catalog could not be accessed for options initialization.`
The CORE catalog cannot be accessed. This action might 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. The SAS installation representative might need to check with IT staff or contact Microsoft technical support for specific windows error codes.

Initialization and Termination Error Messages

If SAS issues error messages during SAS initialization or termination, the SAS log can 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. Also, messages can be written to the SAS console log,

which is a Windows file. Under Windows, the SAS console log is typically located in **`c:\Users\`***user ID*`\AppData`. You can obtain the location and 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**.

Appendix 3

Graphics Considerations under Windows

Using TrueType Fonts with SAS/GRAPH Software 765

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 be printed (other than the default), you must identify its name. To identify TrueType fonts, select **Start** ⇨ **Settings** ⇨ **Control Panel** and double-click the Fonts icon. TrueType fonts have a double 'T' icon or have TrueType beside the font.

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="arial"
  'This is TrueType font arial';
```

Note: You must put TrueType system font names in quotation marks.

For more information about the FONT= option, such as how to use alternate fonts with hard copy devices and how to make an alternate font the default font, see *SAS/GRAPH: Reference*.

Appendix 4

Default Key Settings for Interactive SAS Sessions under Windows

Default Key Definitions under Windows 767
Keyboard Shortcuts within the SAS Main Window 769
Keyboard Shortcuts within the Enhanced Editor 772
Keyboard Shortcuts within Print Preview 776

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 and the Results Viewer. "Keyboard Shortcuts within the Enhanced Editor" on page 772 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 769 .

To browse or change any of the key definitions that are listed in this table, select **Tools** ⇨ **Options** ⇨ **Keys** or issue the KEYS command.

Table A13.1 *Default Key Settings for SAS under Windows*

Key	Default Setting	Key	Default Setting
F1	help		

Key	Default Setting	Key	Default Setting
F2	reshow		
F3	end		
F4	recall		
F5	wpgm		
F6	log	Ctrl + B	LIBNAME
F7	output	Ctrl + D	dir
F8	zoom off; submit	Ctrl + E	clear
F9	keys		
F11	command focus	Ctrl + H	help
		Ctrl + I	options
Shift + F1	subtop		
		Ctrl + K	cut (Program Editor only)
		Ctrl + L	log
Shift + F7	left	Ctrl + M	mark
Shift + F8	right	Ctrl + Q	filename
		Ctrl + R	rfind
Shift + F10	wpopup	Ctrl + T	title
		Ctrl + U	unmark
		Ctrl + W	access (Explorer)
		RMB[1]	wpopup

1 RMB is the right mouse button. 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 A13.2 *Key Settings for the Main SAS Window*

Category	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
	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)

Category	Key Combination	Action
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 insertion point position to the end of the current word
	Ctrl + Backspace	delete from the insertion point position to the start of the current word
	Ctrl + MB1	selects the entire line (clicked line)
	Ctrl + Z	undo previous action
	Ctrl + X	cut selected text
	Ctrl + C	copy selected text to paste buffer
	Ctrl + V	paste text

Category	Key Combination	Action
Window Control	Alt	switch focus to or from the main menu bar
	Alt + F4	exit SAS
	Shift + F5	cascade the windows
	Shift + F4	tile the windows vertically
	Shift + F3	tile the windows horizontally
	Shift + F10	open pop-up menu
	Ctrl + F6	next window
	Ctrl + F4	close the active window
	Ctrl + W	access new SAS explorer window
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 left
	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

Category	Key Combination	Action
	Esc	end docking view resizing without resizing the docking view
Miscellaneous	Alt + Enter	open the Properties dialog box for a selected object 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 A13.3 *Default Keyboard Shortcuts for the Enhanced Editor*

Category	Keyboard Shortcut	Action
Abbreviation	Ctrl + Shift + A	add a new abbreviation
	Alt + F1 + No Selection	bring up word tip
	Esc	hide the current word tip
Code Collapsing	Alt + Ctrl + Number pad -	collapse all tiered blocks
	Alt + Number pad -	collapse current line
	Alt + Ctrl + Number pad +	expand all tiered blocks
	Alt + Number pad +	expand current line
	Alt + Number pad *	toggle expand current line
Command and Macro Support	Ctrl + Shift + M	add or change macros

Category	Keyboard Shortcut	Action
	Ctrl + F1	execute the last recorded macro
	Alt + F8	play a command and macro
	Alt + Shift + R	start and complete macro
Edit	Ctrl + C Ctrl + Insert	copy selection
	Ctrl + X Shift+Delete	cut selection
	Delete	delete current character **Note:** You can delete the entire line by using SHIFT +END to extend the selection to the end of the line and pressing the delete or backspace key.
	Backspace or Shift + Backspace	delete previous character
	Ctrl + Delete	delete to next word start
	Ctrl + Backspace	delete to previous word start
	Enter	insert a carriage return
	Ctrl + V Shift + Insert	paste from clipboard
	Ctrl + Y Alt + Shift + Backspace	redo
	Ctrl + Z Alt + Backspace	undo
Help	place the insertion point within a procedure name and press F1	get Help for a SAS procedure

Category	Keyboard Shortcut	Action
	F1	context Help
Line Markers	F2	go to the next marked line
	Shift + F2	go to the previous marked line
	Ctrl + F2	toggle marker on the current line
Navigation	Ctrl + G	go to line (interactive)
	Ctrl + Page Up Ctrl + Home	move cursor to the top of the file
	Ctrl + Page Down Ctrl + End	move cursor to the bottom of the file
	Down	move cursor down
	Page Down	move cursor down a page
	Left	move cursor left
	Right	move cursor right
	Home	move cursor to beginning of line
	End	move cursor to end of line
	Ctrl + [Ctrl +]	move cursor to matching brace and parentheses
	Alt + [Alt +]	move cursor to matching DO and END keyword
	Alt + Right	move cursor to next case change
	Ctrl + Right	move cursor to next word start
	Alt + Left	move cursor to previous case change

Category	Keyboard Shortcut	Action
	Ctrl + Left	move cursor to previous word start
	Up	move cursor up
	Page Up	move cursor up a page
	Alt + Up	move cursor to the first visible line
	Alt + Down	move cursor to the last visible line
	Ctrl + Down	scroll screen down
	Ctrl + Up	scroll screen up
Option Setting	Insert	toggle insert and overwrite mode
Selection	Shift + Left	extend selection character left
	Shift + Right	extend selection character right
	Shift + Down	extend selection down
	Shift + Page Down	extend selection down a page
	Ctrl + Shift + Home Ctrl + Shift + Page Up	extend selection to beginning of document
	Ctrl + Shift + End Ctrl + Shift + Page Down	extend selection to end of document
	Shift + Home	extend selection to beginning of line
	Shift + End	extend selection to end of line
	Alt + Shift + Right	extend selection to next case change

Category	Keyboard Shortcut	Action
	Alt + Shift + Left	extend selection to previous case change
	Shift + Up	extend selection up
	Ctrl + Shift + Right	extend selection to the next word start
	Shift + Page Up	extend selection up a page
	Ctrl + Shift + Left	extend selection to previous word start
	Ctrl + A	select all
Selection Operations	Ctrl + Shift + W	clean up whitespace characters
	Ctrl + /	comment the selection with line comments
	Ctrl + Shift + L	convert the selected text to lowercase
	Ctrl + Shift + U	convert the selected text to uppercase
	Tab + Selection	tab selection
	Ctrl + Shift + /	undo the comment
	Shift + Tab + Selection	left tab selection

Keyboard Shortcuts within Print Preview

You can use the keyboard shortcuts in the following table in the Print Preview window.

Table A13.4 *Keyboard Shortcuts for the Print Preview Window*

Action	Keyboard Shortcut
Next page or Page Down	Alt + N
Previous page or Page Up	Alt + P
Zoom	Alt + Z
Help	Alt + H
Print	Alt + R
Close the window	Alt + C Alt + F4

Appendix 5

Using EBCDIC Data on ASCII Systems

About EBCDIC and ASCII Data **779**
Overview of EBCDIC and ASCII Data Representation 779
EBCDIC File Structures 780
ASCII File Structure 781
Numeric Values 782

Moving Data from EBCDIC to ASCII Systems **782**
Overview of Accessing EBCDIC Data on ASCII Systems 782
Example of Incorrect Conversion of Packed-Decimal Numeric Data 783
Convert EBCDIC Files with Fixed-Length Records 784
Convert EBCDIC Files with Variable-Length Records 785
Read EBCDIC Data from Structured COBOL Files 789

Moving Data from ASCII to EBCDIC Systems **790**
Overview 790
Using FTP to Write Files Directly 790
Using the dd Command to Convert and Copy a File 792
Using the iconv Command to Convert a Text File 793

About EBCDIC and ASCII Data

Overview of EBCDIC and ASCII Data Representation

Extended Binary Coded Decimal Interchange Code (EBCDIC) is an 8-bit character encoding method for IBM mainframe machines. *American Standard Code for*

Information Interchange (ASCII) is a 7-bit character encoding method for most other machines, including Windows, UNIX, and Macintosh machines.

Hexadecimal characters are used to represent one byte or eight bits of data. In a binary system, each bit can have the value 0 or 1. An aggregation of four bits can therefore take on 16 (2^4) possible values. This means that two hexadecimal characters can be used to represent one byte of data. In the EBCDIC and ASCII encoding methods, each character is represented by two hexadecimal characters. (This pertains primarily to Western language, single-byte encoding methods. There are other encoding methods that store a single character in two bytes of storage, such as encoding methods that are used for Japanese or Korean data.)

Each encoding method represents the same data differently, as shown in the following examples:

- On an EBCDIC system, the digit 4 is represented by the hexadecimal value 'F4'x. On an ASCII system, the digit 4 is represented by the hexadecimal value '34'x.
- On an EBCDIC system, the hexadecimal value '50'x represents the symbol &. On an ASCII system, the same hexadecimal value represents the letter P.

When SAS reads a file, it expects the data in the file to be in the encoding that matches the ENCODING= option for the SAS session. For example, on a Windows machine, the default encoding for a single-byte SAS session with a US English locale is LATIN1. SAS expects the data in a file on that Windows machine to use a LATIN1 encoding. However, if a file originates on an EBCDIC machine and it is stored on a Windows machine, then SAS would misinterpret the data from this file if no other encoding information is provided. For this reason, specific steps must be performed to convert data that originates on an EBCDIC system before it can be used on an ASCII system (for example, the Windows machine). Here are the two main methods to make EBCDIC data available on an ASCII system:

- On the ASCII system, read the data directly from the EBCDIC system.
- Use an FTP program to move the data, with or without any conversion of the data.

EBCDIC File Structures

When you decide how to move data from an EBCDIC system to an ASCII system, consider the structure of the EBCDIC source file. On EBCDIC systems, you might have files with fixed-length records or files with variable-length records. Either type of file contains a header with information about the file. The header includes a Record Format attribute that indicates whether the records are fixed length or variable length. The header for a file with fixed-length records includes a Logical Record Length attribute that indicates the length of each record in bytes.

In SAS, the Record Format attribute corresponds to the RECFM= option in a FILENAME statement. To access a file with fixed-length records, specify RECFM=F. To access a file with variable-length records, specify RECFM=V. Similarly, the Logical Record Length attribute corresponds to the LRECL= option.

The Logical Record Length attribute in the header for a file with variable-length records indicates the maximum record length. Each record in a file with variable-length records begins with a *record descriptor word (RDW)*. The RDW is a 4-byte binary integer field. The first two bytes of the RDW indicate the length of the current record. The last two bytes of the RDW contain information that is used by the operating system. The length of the record includes the four bytes of the RDW at the beginning of the record. Because the length of each record is specified in an EBCDIC file (either in the header or in the RDW), there are no end-of-record indicators in EBCDIC files.

A file with variable-length records also contains *block descriptor words (BDWs)*. Like the RDW, the BDW is a 4-byte, binary integer field. The first two bytes indicate the block size, and the last two bytes are used by the operating system. Each block can contain multiple records. If the block size is not specified when the file is created, the default block size is the logical record length plus 4. Otherwise, the size of a block is the number of bytes that are contained in the block. This value is the sum of the record lengths in the block (obtained from the RDWs) plus 4 (the length of the BDW).

ASCII File Structure

On ASCII systems, a file does not contain a header with information about the file, such as record format or lengths. The RECFM attribute for ASCII files is variable (RECFM=V), and the record length (LRECL) is unlimited. Instead of defining record lengths like EBCDIC files do, ASCII files use end-of-record indicators to flag the end of a record. On a Windows machine, the end-of-record indicators are the carriage return (CR) and line feed (LF) characters. On a UNIX machine, an LF indicates the end of a record. On a Macintosh machine, a CR indicates the end of a record. Other types of machines use different combinations of characters to identify the end of record. For all ASCII machines, the hexadecimal value for CR is '0D'x, and the hexadecimal value for LF is '0A'x.

When SAS reads a file from disk on an ASCII machine, default values for some file attributes must be used because these attributes are not defined. The default RECFM value is V (variable-length record), and the default LRECL value is 32767. This means that SAS scans the input from an ASCII file, parses the data into variable values based on the INPUT statement, and looks for an end-of-record indicator. If the end of a record is not found within the specified number of characters (based on LRECL), then SAS truncates the record and prints a message in the log. For example, suppose LRECL is set to 256, and there is a record that is 300 characters. SAS reads the first 256 characters based on the INPUT statement, and then discards the last 44 characters. A message in the log states that “One or more lines have been truncated.” You can override the current LRECL value using the LRECL= option in the INFILE statement.

Numeric Values

When stored as character data, the decimal digits 0 through 9 each occupy one byte of storage. One 8-bit byte includes two 4-bit *nibbles*. Each nibble can have 16 (2^4) possible values. The first nibble is the *high-order nibble*, and the second is the *low-order nibble*. In EBCDIC and ASCII systems, the high-order nibble has a standard value. Decimal digits are represented in EBCDIC with a high-order nibble of F. Decimal digits are represented in ASCII with a high-order nibble of 3. This means that in an EBCDIC system, the digits 0 through 9 are represented by the hexadecimal values 'F0'x through 'F9'x. In an ASCII system, the digits 0 through 9 are represented by the hexadecimal values '30'x through '39'x. This encoding method treats decimal digits as characters.

As an alternative to storing decimal digits as characters, there are other encoding methods that can be used on an EBCDIC system. For example, a packed-decimal encoding method represents two decimal digits in one byte of storage. A zoned-decimal encoding method represents one decimal digit in one byte of storage, and the sign of the entire value is included within one byte of storage. (The byte that stores the decimal digit and the sign of the entire value can be either the first byte or the last byte, depending on the type of machine.)

You must know the numeric encoding that is used on the source EBCDIC system so that the source data is interpreted correctly on the ASCII system. For SAS, this means that you must specify the correct informats to use for numeric data.

Moving Data from EBCDIC to ASCII Systems

Overview of Accessing EBCDIC Data on ASCII Systems

There are several ways to access EBCDIC data on an ASCII system. For example, some ASCII machines have peripheral devices that can read 3480 or 3490 cartridge tapes that are created on an EBCDIC system. These devices can read the data directly from a tape into an application on an ASCII machine. Alternatively, these devices can copy data from a tape and store it on the ASCII machine's hard drive.

A more common method of moving and converting data is to use an FTP program to transfer the data. By default, most FTP programs convert EBCDIC data into ASCII when transferring data. If the source data contains only character data (including

digits that are encoded as characters), this is the recommended method. During the conversion process, the FTP program creates the appropriate end-of-record indicators for the ASCII system. After conversion, you can use an INFILE statement to access the newly created file on the ASCII system. Use an INPUT statement to specify the correct informat values to use when reading the data in the file.

Note: Even when all of the EBCDIC source data is encoded as character data, there might be some characters that are not interpreted correctly during conversion. The correct interpretation of these characters depends on the encoding method that is used on the EBCDIC machine. As a best practice, verify that your data was converted correctly by viewing the data that SAS reads from a converted file.

When an EBCDIC file contains numeric data that is not encoded as character data, such as when a packed-decimal or zoned-decimal encoding method is used, the default FTP conversion does not work correctly. Some numeric data can resemble standard character data. In this case, FTP conversion incorrectly assigns ASCII characters to EBCDIC numeric data. For more information, see "Example of Incorrect Conversion of Packed-Decimal Numeric Data".

Note: There is no way to correctly convert packed-decimal encoded data from EBCDIC into ASCII. Other methods to convert the data must be used if a packed-decimal, zoned-decimal, or other numeric encoding method is used on the EBCDIC system. For more information, see "Convert EBCDIC Files with Variable-Length Records" on page 785.

In some instances, a byte of EBCDIC data might be interpreted in ASCII as an end-of-line flag or end-of-file flag. If SAS is reading a file with variable-length records when one of these hexadecimal values is encountered, then you might observe unintended results. Depending on the expected data values based on specified informats, you might observe anything from invalid data errors to unexpected termination of the DATA step.

Example of Incorrect Conversion of Packed-Decimal Numeric Data

This example demonstrates the problems that can result when you convert packed-decimal numeric data as if it were encoded as character data. Suppose an EBCDIC data file contains the numeric value 505, stored as a packed-decimal value ('505C'x). If you looked at the file with an EBCDIC file browser or editor, you would see the characters '&*'. This is because '50'x corresponds to '&' and '5C'x corresponds to '*'. The FTP program interprets the '&' character and converts it to the ASCII value '26'x. The FTP program converts the '*' character to the ASCII value '2A'x, and the resulting converted value is '262A'x. The correct packed-decimal value in ASCII should be '000505'x. Because the input data does not conform to the expected packed-decimal informat, SAS prints an error to the log that states that the data is invalid. Each time invalid data is encountered, SAS writes an error

to the log, and prints the contents of the input buffer and the corresponding DATA step variables.

Table A14.1 *Incorrect Conversion of Packed-Decimal Numeric Data*

Step	Action	Value
1	FTP program reads the EBCDIC packed-decimal numeric value '505'.	'505C'x
2	FTP program interprets the value as standard EBCDIC characters.	&*
3	FTP program converts to standard ASCII hexadecimal characters.	'262A'x
4	SAS flags the data as invalid because packed-decimal numeric data is expected (based on the specified informat value).	???

Convert EBCDIC Files with Fixed-Length Records

FTP the File in Binary

When you convert an EBCDIC file with fixed-length records, use FTP to transfer the file in binary. Then, with a FILENAME or INFILE statement, specify RECFM=F, and assign the same value to LRECL that the file has in the EBCDIC system. Use the formatted input style with the following informats:

- $EBCDIC*w*. for character input data
- S370F*xxxw.d* for numeric input data

> **Note:** There are many S370F*xxxw.d* informats. Select those informats that match the type of data that you have. For more information, see *SAS Formats and Informats: Reference* for SAS 9.3 and higher.

Because you are transferring the source file in binary, there is no processing to add end-of-record indicators. For this reason, you must specify the exact number of bytes that are specified for the source file in the EBCDIC system. If there are bytes

in the source file that would be interpreted as end-of-record indicators or end-of-file indicators in an ASCII context, SAS treats those bytes simply as data.

Example: Convert an EBCDIC File with Fixed-Length Records into an ASCII File

The following code reads a file, fixed.txt, that was previously transferred via FTP in binary from an EBCDIC system to an ASCII system. The source file has fixed-length records that are 60 bytes long. Based on the informat in this example, the last three bytes in each record contain numeric data that was stored using the packed-decimal encoding method.

```
filename test1 'c:\fixed.txt' recfm=f lrecl=60;
data one;
infile test1;
input @1  name  $ebcdic20.
      @21 addr  $ebcdic20.
      @41 city  $ebcdic15.
      @56 state $ebcdic2.
      @58 zip   $s370fpd3.;
run;
```

Convert EBCDIC Files with Variable-Length Records

Overview of Converting EBCDIC Files with Variable-Length Records

When you convert an EBCDIC file with variable-length records, you can use an FTP program. The FTP program removes BDWs and RDWs and adds end-of-record indicators that are expected by the ASCII system. The data in the file is converted from EBCDIC to ASCII. If all of the data in the EBCDIC file is encoded as characters, then this process typically works correctly.

Note: Even when all of the EBCDIC source data is encoded as character data, there might be some characters that are not interpreted correctly during conversion. The correct interpretation of these characters depends on the encoding method that is used on the EBCDIC machine. As a best practice, verify that your data was converted correctly by viewing the data that SAS reads from a converted file.

When an EBCDIC file contains numeric data that is not encoded as character data, such as when a packed-decimal or zoned-decimal encoding method is used, the default FTP conversion does not work correctly. For more information, see

"Overview of Accessing EBCDIC Data on ASCII Systems" on page 782. To prevent misinterpretation of data during conversion, transfer the file in binary via FTP without converting the data to an ASCII encoding. When the data is transferred in binary and is not converted, be aware that the BDW and RDW information is removed automatically. This removes information that SAS needs to read the data successfully.

Read Files Directly from the EBCDIC System

If you have direct access between the ASCII machine and the EBCDIC machine, then the best practice is to read the file directly. Direct access is enabled via a peripheral device on the ASCII machine that can read an EBCDIC tape. You can access the file via the FTP access method in a FILENAME statement. There are several advantages to this method of accessing EBCDIC data:

- file preprocessing is not required
- copying the source file is not required
- FTP access method works for fixed-length and variable-length records
- DATA step processing works as expected

The main disadvantage is that this method requires more time for processing because you are accessing the data remotely.

This method of accessing EBCDIC data applies if you have a 3480 or 3490 cartridge tape reader attached to your ASCII machine. In this case, you do not need to preprocess the file on an EBCDIC machine. You can read it directly from the tape by setting RECFM=S370VB and using the $EBCDIC*w*. and S370F*xxxw.d* informats.

In a FILENAME statement, specify the FTP access method and the source filename, and provide values for the HOST=, USER=, and PASS= options. The HOST= option specifies the name of the EBCDIC machine, USER= specifies the user account that you use to log on, and PASS= specifies the password that you use to log on. The FTP access method uses an FTP program on the ASCII machine to open a connection between the ASCII machine and the EBCDIC machine. The SAS system connects to and logs on to the mainframe machine with the specified user account and password. The FTP program transfers the file.

Note: If you specify the PASS= option, the password is saved as text in your SAS program. The password is not visible in the SAS log. As an alternative to the PASS= option, you can specify the PROMPT option and provide a password at the prompt when you execute the SAS program.

For EBCDIC files with variable-length records, you must also specify the S370V and RCMD= options. The S370V option indicates that the records in the source file have variable lengths. For the RCMD= option, specify RCMD="SITE RDW" to indicate that the FTP process should keep the RDW information during the file transfer.

If you experience connection problems to the EBCDIC machine, you can add the DEBUG option to see the informational messages that are sent to and from the FTP server.

Example: Read an EBCDIC Source File Directly with the FTP Access Method

This example shows how to read an EBCDIC file with variable-length records directly from an EBCDIC machine using the FTP access method. The user is prompted for her MVS logon password. The ZIP code is entered as a 5-digit EBCDIC number, represented by one digit per byte. The comments section is varying in length up to 200 characters. After the data is read, it is printed to verify the contents of the data set.

```
filename test1 ftp "'SASEBCDIC.VB.TEST1'" host='MVS' user='SASEBCDIC'
PROMPT
          s370v rcmd='site rdw';
data one;
infile test1;
input @1  name     $ebcdic20.
      @21 addr     $ebcdic20.
      @41 city     $ebcdic10.
      @51 state    $ebcdic2.
      @54 zip      s370ff5.
      @60 comments :$ebcdic200.;
run;

proc print;
run;
```

Reformat an EBCDIC File with Variable-Length Records with IEBGENER

Suppose that you do not have direct access between the ASCII machine and the EBCDIC machine. That is, you do not have a peripheral device that reads EBCDIC data on the ASCII machine. In this situation, you can convert the data by reformatting the file on the mainframe machine. By changing the format of the file, you prevent the FTP program from removing the RDW information that SAS requires to read the data correctly. After you reformat the file, you can transfer the file in binary to the ASCII machine.

To reformat the source file, use the IEBGENER program on the EBCDIC machine. Use this program to make an exact copy of the file with altered header information. Specifically, use IEBGENER to change the RECFM value from V (variable-length records in blocks) to U (undefined record length and unblocked). After making this change, the FTP program no longer removes the RDW information during the file transfer.

When you run the IEBGENER program, in addition to the required arguments, specify the following overrides:

```
SYSUT1 DCB=(RECFM=U,BLKSIZE=32760)
SYSUT2 DCB=(RECFM=U,BLKSIZE=32760) DISP=(NEW,CATLG)
```

Note: Do not use the original values of RECFM and BLKSIZE for SYSUT1.

Transfer the new version of the file in binary using an FTP program on the ASCII machine. In SAS, use a FILENAME or INFILE statement to read the transferred file. Set the options appropriately.

- Set the RECFM= option to S370V if the record format for the original file was variable (RECFM=V). Set the RECFM= option to S370VB if the record format for the original file was variable and blocked (RECFM=VB). By specifying the RECFM= option as S370V or S370VB, you tell SAS to process the RDW information for each record and enter the correct number of bytes for each record.
- Specify the same value for the LRECL= option that is in the original file. If you do not specify a value for the LRECL= option, SAS uses the default LRECL value (32767). Using the default value could cause SAS to truncate data records if they are longer than the default LRECL value.

Use the formatted input style with the informats that are described in "FTP the File in Binary" on page 784.

Example: Read a File with Modified Header Data

This example reads a file that was generated from an EBCDIC file with a header that was modified to change the file format. The modified file was transferred to an ASCII machine for SAS processing. For more information, see "Reformat an EBCDIC File with Variable-Length Records with IEBGENER" on page 787.

The TRUNCOVER option is included in the INFILE statement because the Comment variable can be up to 60 characters (but it is likely shorter). Without the TRUNCOVER option, the INPUT statement could attempt to read past the end of the record. Data from the next record would continue to be assigned to the Comment variable until the variable was full. The LRECL= option is not specified because the default value is sufficient to handle the longest record in the file. After the data is read, it is printed to output for verification.

```
filename test1 'c:\vbtest.xfr' recfm=s370vb;
data one;
infile test1 truncover;
input @1  name    $ebcdic14.
      @15 addr    $ebcdic18.
      @33 zip     s370ff5.
      @38 comment $ebcdic60.;
run;

proc print;
run;
```

Read EBCDIC Data from Structured COBOL Files

About Structured COBOL Files

A structured COBOL file is generated using an OCCURS DEPENDING ON clause. This type of file has variable-length records. And, when the file is transferred via FTP in binary, there is no BDW or RDW information. Each record is divided into three parts: a record header (a fixed-length portion of the record), an index variable, and one or more data segments. The documentation for the file provides the length of the record header, the index variable, and a data segment. The record header is the same length for each record. It contains information that pertains to all of the data segments that follow. The index variable provides the number of data segments for the current record. The remainder of the record contains the data segments.

Because of the structure of the records, SAS is able to read the data in these files. The length of a record is the sum of the header length, the index length, and the product of the index value and the size of each data segment. For each data segment, SAS reads the segment, and then writes a copy of the header and the current data segment to a new observation in a SAS data set.

When you read a structured COBOL file, specify RECFM=N in your FILENAME statement. This tells SAS that you are reading a stream of data that does not conform to a typical file structure. Any restrictions to record length are ignored when SAS reads a data stream because SAS does not attempt to buffer the input. SAS writes a statement to the SAS log to notify you that SAS reads a data stream as unbuffered when RECFM=N.

SAS reads an entire structured COBOL file as a single, long record. Therefore, if you need to skip some data or move past a space, you must use relative column pointers in your INPUT statement. Line holders are ignored because the contents of the file are treated as a single input record. The @*column* pointers do not work for these files.

CAUTION

Do not use @*column* pointers when you specify RECFM=N. Using @*column* pointers initiates an infinite loop in which SAS reads and writes the same data repeatedly until you halt the program or until no more disk space is available.

Example: Read Data from a Structured COBOL File

In this example, an EBCDIC file was transferred via FTP in binary without first processing the file using IEBGENER. The record header (fixed-length) portion of each record is 59 bytes in length and contains a combination of character and

numeric data. The index variable is two bytes. There is another space (one byte) to separate the index variable from the remainder of the record. The data segment portion of the record consists of one or more repeats of 13 bytes in length. Each repeat contains a combination of character and numeric data.

```
filename test1 'c:\VB.TEST' recfm=n;

data one;
infile test1;
input name $ebcdic20. addr $ebcdic20. city $ebcdic10. st $ebcdic2. +1
      zip s370ff5. +1 idx s370ff2. +1;
do i = 1 to idx;
   input cars $ebcdic10. +1 years s370ff2. ;
   output;
   if i lt idx then input +1 ;
end;
run;
```

Moving Data from ASCII to EBCDIC Systems

Overview

There are several ways to transcode ASCII data to EBCDIC:

- Use FTP to write files (data) directly.
- Use the `dd` command.
- Use the `iconv` command.

Using FTP to Write Files Directly

Overview of Using FTP to Write Files Directly

FTP automatically performs the conversion when the type of file is specified as text (instead of binary). When you have direct access between the ASCII machine and EBCDIC machine, the best practice is to read the file directly. Direct access is enabled via a peripheral device on the ASCII machine that can read an EBCDIC tape. You can access the file via the FTP access method in a FILENAME statement. There are several advantages to this method of accessing EBCDIC data:

- No file preprocessing is required.
- You do not need to copy the source file.
- The FTP access method works for fixed-length and variable-length records.
- DATA step processing works as expected.

This method of accessing EBCDIC data applies if you have a 3480 or 3490 cartridge tape reader attached to your ASCII machine. In this case, you do not need to preprocess the file on an EBCDIC machine. You can read it directly from the tape by setting RECFM=S370VB and using the $EBCDICw. and S370Fxxxw.d informats.

In a FILENAME statement, specify the FTP access method and the source filename, and provide values for the HOST=, USER=, and PASS= options. The HOST= option specifies the name of the EBCDIC machine, USER= specifies the user account that you use to log on, and PASS= specifies the password that you use to log on. The FTP access method uses an FTP program on the ASCII machine to open a connection between the ASCII machine and the EBCDIC machine. The SAS system connects to and logs on to the mainframe machine with the specified user account and password. The FTP program transfers the file.

Example: Writing an ASCII File from SAS to z/OS

This example shows how to write an ASCII file with fixed-length records directly to z/OS using the FTP access method. This example uses the fixed.txt file created in "Example: Convert an EBCDIC File with Fixed-Length Records into an ASCII File" on page 785. CD specifies the z/OS file to create. No password is specified. Instead, the user is prompted for the logon password.

Note: This example is run in a SAS session on UNIX.

```
filename unixin ftp host='remote-ftp-host' user='sluggo'
         cd='path-to-zOS-files' recfm=f lrecl=60 binary PROMPT;
data _null_;
      infile 'path/fixed.txt';
      input;
      put _infile_;

run;
```

Example: Reading an ASCII File from SAS on z/OS

This example shows how to read an ASCII file that was previously transferred using FTP. Because the type of file is text, FTP automatically performs the conversion.

Note: This example is run in a SAS session on z/OS.

```
filename unixin '/net/bin/u/<user>/sample.txt' encoding=latin1;
```

```
data _null_;
      infile unixin;
      input;
      put _infile_;
run;
```

Example Code A14.1 *Log Output*

```
NOTE: The infile UNIXIN is:
      File Name=/net/bin/u/<user>/sample.txt,
      Access Permission=-rwxr-xr-x,Number of Links=1,
      Owner Name=<user>,Group Name=R@D,File Size=45,
      Last Modified=Jan 19  2000

This is a test.
Another line.
End of file.
```

Using the dd Command to Convert and Copy a File

About the dd Command

The `dd` command reads the InFile parameter or standard input, performs the specified conversion, and then copies the converted data to the OutFile parameter or standard output. The input block size and output block size can be specified to take advantage of raw physical I/O.

Use the cbs parameter value if you are specifying the `block`, `unblock`, `ascii`, `ebcdic`, or `ibm` conversion value. If an `unblock` or `ascii` value is specified, then the `dd` command performs a fixed-length to varying-length conversion. Otherwise, it performs a varying-length to fixed-length conversion. The cbs parameter value determines the fixed length.

CAUTION
If the specified cbs parameter value is smaller than the smallest input block, the converted block is truncated.

After it finishes, the `dd` command reports the number of whole and partial input and output blocks. For more information about the `dd` command, see the `dd` manual page on your system.

dd Command Exit Status

The `dd` command returns the following exit values:

Table A14.2 *Exit Status Values for the dd Command*

Item	Description
0	The input file was copied successfully.
>0	An error occurred.

Examples: dd Command Conversion

Here are two simple examples:

- To convert an ASCII text file to EBCDIC, enter the following:

  ```
  dd if=text.ascii of=text.ebcdic conv=ebcdic
  ```

  This command converts the text.ascii file to EBCDIC representation and stores the EBCDIC version in the text.ebcdic file.

  When you specify the conv=ebcdic parameter, the `dd` command converts the ASCII ^ (circumflex) character to an unused EBCDIC character (9A hexadecimal) and the ASCII ~ (tilde) character to the EBCDIC ^ character (NOT symbol).

- To use the `dd` command as a filter, enter the following:

  ```
  ls -l | dd  conv=ucase
  ```

  This command displays a long listing of the current directory in uppercase.

  The performance of the `dd` command and `cpio` command in the IBM 9348 Magnetic Tape Unit Model 12 can be improved by changing the default block size. To change the block size, use the `chdev` command as follows:

  ```
  chdev -l Device_name -a block_size=32k
  ```

Using the iconv Command to Convert a Text File

About the iconv Command

Use the `iconv` command to convert the encoding of a text file. Use one the following examples of syntax:

```
iconv -f FromCode -t ToCode FileName
```

```
iconv -l
```

For more information about the syntax and parameters for the `iconv` command, see the `iconv` manual page on your system.

iconv Command Exit Status

The `iconv` command returns the following exit values:

Table A14.3 *Exit Status Values for the iconv Command*

Item	Description
0	Input data was successfully converted.
1	The specified conversions are not supported, the input file cannot be opened or read, or there is a usage-syntax error.
2	An unusable character was encountered in the input stream.

Examples: iconv Command Conversion

Here are two simple examples:

- To convert the contents of the mail.x400 file from code set IBM-850 and store the results in the mail.local folder, enter the following:

```
iconv -f IBM-850 -t ISO8859-1 mail.x400 > mail.local
```

- To convert the contents of a local file to the mail interchange format and send mail, enter the following:

```
iconv -f IBM-943 -t fold7 mail.local > mail.fxrojas
```

24

SCL Methods for Automating OLE Objects under Windows

Summary of OLE Class Methods **795**

Dictionary **796**
COMPUTE Method 796
_DISABLE_DEFAULT_ACTION_ Method 798
_DO_ Method 798
_ENABLE_DEFAULT_ACTION_ Method 799
EXECUTE Method 799
_GET_EVENT_ Method 800
_GET_PROPERTY_ Method 801
_GET_REFERENCE_ID_ Method 801
_GET_TYPE_ Method 802
_IN_ERROR_ Method 803
NEW Method 804
_SET_PROPERTY_ Method 804
UPDATE Method 805

Summary of OLE Class Methods

Table A6.4 on page 796 contains a list of SCL methods that you can use with object linking and embedding (OLE) and indicates which of the OLE classes they apply to.

Table A15.1 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
_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.

Dictionary

COMPUTE Method

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*);

Details

Table A15.2 *OLE Arguments*

Argument	Character (C) or Numeric (N)	Description
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.

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.

Example

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. It 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_ Method

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_ Method

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*>);

Details

Table A15.3 *OLE Arguments*

Argument	Character (C) or Numeric (N)	Description
in-OLE-method	C	specifies the OLE method name.
in-parm	C or N	provides a parameter to the OLE method.

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_ Method

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 Method

Executes an OLE verb for the object.

Syntax

CALL NOTIFY(*OLE-object-name*,'_EXECUTE_',*in-verb*<,*in-verb*...*in-verb*>);

Details

Table A15.4 OLE Arguments

Argument	Character (C) or Numeric (N)	Description
in-verb	C	specifies the OLE verb to execute.

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_ Method

Returns the name of the last OLE control event received.

Syntax

CALL NOTIFY(*OLE-object-name*,'_GET_EVENT_',*out-event*);

Details

Table A15.5 OLE Arguments

Argument	Character (C) or Numeric (N)	Description
out-event	C	contains the returned name of the last OLE control event received.

This method returns the name of the event that the specified OLE control most recently sent.

_GET_PROPERTY_ Method

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*);

Details

Table A15.6 *OLE Arguments*

Argument	Character (C) or Numeric (N)	Description
in-OLE-property	C	specifies the name of the OLE property.
out-property-value	C or N	contains the returned value of the property.

The _GET_PROPERTY_ method is used to get the value of a property of an automation object.

_GET_REFERENCE_ID_ Method

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*);

Details

Table A15.7 OLE Arguments

Argument	Character (C) or Numeric (N)	Description
out-refid	C	contains the returned reference identifier.

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_ Method

Returns the object's type.

Syntax

CALL NOTIFY(*OLE-object-name*,'_GET_TYPE_',*out-type*);

Details

Table A15.8 *OLE Arguments*

Argument	Character (C) or Numeric (N)	Description
out-type	C	contains the returned object type.

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_ Method

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*>);

Details

Table A15.9 *OLE Arguments*

Argument	Character (C) or Numeric (N)	Description
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.

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 before 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 Method

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_ Method

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*);

Details

Table A15.10 *OLE Arguments*

Argument	Character (C) or Numeric (N)	Description
in-OLE-property	C	specifies the OLE property name.
in-value	C or N	contains the value to assign to the OLE property.

The _SET_PROPERTY_ method assigns a value to a property of an automation object.

UPDATE Method

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*>);

Details

Table A15.11 *OLE Arguments*

Argument	Character (C) or Numeric (N)	Description
in-hservice	C	specifies the name of the HSERVICE entry to use to update the object.

The _UPDATE_ method re-creates 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 process 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.Service object:

```
length refid $ 30;
call notify('obj1','_update_',
            'sasuser.examples.sound1.hservice');
```

25

Cleanwork Utility

Cleanwork Utility 807
Scheduling the Cleanwork Utility with Microsoft Task Scheduler 808
Dictionary 809
Cleanwork Utility Window 809

Cleanwork Utility

The Cleanwork utility searches and cleans temporary SAS files and SAS utility directories. The Cleanwork utility, cleanwork.exe, is a command-line utility and is installed as part of Base SAS in the SASROOT directory. The Cleanwork utility replaces the SAS Disk Cleanup Handler Utility.

Cleanwork searches the specified directory or volume for SAS work directories with the following formats. If the process ID is no longer in use and the nodename matches the host name of the machine that created the directory, cleanwork deletes the entry. The contents of the directory are removed.

#TD<pid>, _TD<pid>, _TD<pid>_<nodename>_

where

- *#TD* is the SAS V6.X prefix
- *_TD* is the SAS V7+ prefix
- *<pid>*is the process ID
- *nodename* is the host name of the machine that created the directory.

The Cleanwork utility also removes SAS utility directories that have the following format:

SAS_util*NNNNPPPPPPPP_nodename*

where

- *NNNN* is a unique random number
- *PPPPPPPP* is the hexadecimal representation of the process ID
- *nodename* is the host name of the machine that created the directory.

Note: The Cleanwork utility is available beginning with SAS 9.4M2.

Scheduling the Cleanwork Utility with Microsoft Task Scheduler

Cleanwork can be scheduled to run automatically at a specific time with any of the options specified in the arguments section. Follow these instructions to schedule Cleanwork from the Microsoft Task Scheduler:

1. Schedule cleanwork to run with the desired options. Refer to the arguments section for the options descriptions.
   a. Select **Start** and type *Task Scheduler*.
   b. Select **Create Basic Task** from the Actions window.
   c. Enter a name and select **Next**.
   d. Select the frequency and then select **Next**.
   e. Select a starting date and time and then select **Next**.
   f. Select **Start a Program** and then select **Next**.
   g. Enter the location of cleanwork.exe.
   h. Enter the options (for example, /v volume) in the **Add arguments** field.
   i. Select **Next** and then **Finish**.
   j. Right-click on the new task and then select **Properties**.
      1. Select the **Change User or Group** button, in the Security options box.
      2. Enter *SYSTEM* in the Enter the object name to select window, and then select **Check Names**. If you do not set user account as SYSTEM, then the taskeng.exe window is displayed. If SYSTEM is set, then the taskeng.exe window is not displayed.
2. Repeat step 1 to create another configuration for the same program. For example, the user might want to scan the volume once a week, but wants to clean the logs every day or the opposite process. This step is optional.

3 Create a batch script with the cleanwork.exe path and output–redirection options, if you want to review the output at a later date. This action is a Windows Task Scheduler requirement. For example, create a cleanwork.bat script with the following command:`cleanwork.exe /v c: /verbose >c:\temp\cleanwork.log` This step is optional.

Dictionary

Cleanwork Utility Window

Searches and cleans temporary SAS files.

CAUTION: **Any <module name>.PID.DATE.log files in the AppData\Roaming\SAS\LOGS directory that a user wants to save should be relocated to another directory.**

CAUTION: **Cleanwork searches and deletes _TD#### directories. The user should verify that no third-party applications use the same template for directories before running the cleanwork utility.**

Syntax

cleanwork [/userlog] | [/alllogs] [/v volume(s)] [/d dir(s)] [/list] [/verbose]

Arguments

/userlog
Cleans SAS log files in current users AppData\Roaming\SAS\LOGS directory.

/alllogs
Cleans all users AppData\Roaming\SAS\LOGS directories.

CAUTION
This option requires administrator privileges.

/v volume
specifies the volume to search for removal of SAS directories

Note: When specifying more than one volume, separate them with spaces.

/d dir(s)
Cleans all SAS directories in the specified directory(s) where dir(s) are Windows or UNC paths to one or more directories.

Note: Paths need to be separated by a space (for example, /d path1 path2...).

/list
Lists files to be deleted. This option does not delete files or folders.

/verbose
Shows a trace of the program's activity.

Examples

Example 1

```
cleanwork /d c:\Temp
```

```
Cleans all SAS Work and Utility directories located in the c:\Temp directory
hierarchy.
```

Example 2

```
cleanwork /d c:\Temp /userlog
```

```
Cleans all SAS Work and Utility directories located in the c:\Temp directory
hierarchy and all log files in the \AppData\Roaming\SAS\LOGS directory.
```

Example 3

```
cleanwork /v c: u:
```

```
Cleans all SAS Work and Utility directories located on the c: and u: volumes.
```

Example 4

```
cleanwork /alllogs
```

```
Scans all users for a SAS\LOGS directory and cleans all log files found. This
option will require administrator privileges to remove other users' files.
```

Example 5

```
cleanwork /d c:\Temp /list
```

```
List all SAS Work and Utility directories located in the c:\Temp directory
hierarchy to be deleted.
```

Example 6

```
cleanwork /v c: >cleanwork.log
```

```
Cleans all SAS Work and Utility directories located on the c: volume and dumps
the output to the file cleanwork.log instead of printing it out to the screen.
```

Note: Do not use this utility with Windows task scheduler. Instead, write a script that includes the command and set the task scheduler to execute the script.

26

SASIOTEST Utility

Sasiotest Utility 813
Best Practices and Considerations 814
Acceptable SAS I/O Performance 815
Dictionary 815
SASIOTEST Command 815

Sasiotest Utility

When troubleshooting performance problems, you need to know the throughput rates of any file system that SAS uses. The Sasiotest utility performs the following actions:

- measures the I/O behavior of the system under defined loads
- opens files and reads and writes data, similar to SAS I/O
- creates and writes files to the file system that is being tested, and reads them to test Write and Read performance
- writes an output file that captures elapsed real time and I/O rate, expressed as megabytes per second
- reads random pages
- can run several I/O tests concurrently to overload the file system and determine its performance

The following information relates to how the Sasiotest utility operates on Windows:

- runs on 32-bit and 64-bit Windows systems
- is delivered with SAS and is installed in the SASHOME directory at `\SASHome\SASFoundation\9.4\sasiotest.exe`

- is a stand-alone program, and can be launched from a command prompt window
- uses the Windows APIs Writefile(), Readfile(), and Closefile()
- supports the Readfilescatter() and Writefilescatter() APIs for Scatter-Gather I/O testing outside the file cache
- can be copied and used as a stand-alone executable on any system with or without SAS
- can be copied and used on a machine that has previous SAS releases

For information about how SAS performs I/O and the minimum I/O recommendations for SAS file systems, see the SAS papers available at Tuning guidelines and best practices for your hardware infrastructure (systems, storage, virtualization, file systems, and operating systems). The papers outline recommended I/O metrics for file systems that support SAS deployments, and they identify guidelines that can help tune I/O characteristics for optimal SAS performance. If you are testing your file system throughput to solve an existing performance problem, see especially the following papers about performance-problem resolution using SAS logs in combination with host monitors. Select **Performance monitoring and troubleshooting** to access the following papers:

- *Solving SAS Performance Problems: Employing Host Based Tools*
- *A Practical Approach to Solving Performance Problems with SAS*
- *SAS Performance Monitoring - A Deeper Discussion*

Best Practices and Considerations

Consider the following concepts before beginning I/O throughput tests:

- SAS performs the Write and Read operations with the Windows file cache. The file sizes for the Write and Read tests should be larger than your host system file cache. They can be larger than the RAM in your host system. For example, if you run your test with a 1GB file size, and your system has 5GB of cache, then the test processes limited disk I/O on the Read test. The file that is written remains in the system file cache, and the subsequent Read operation reads it from cache instead of from disk. If the Read tests are satisfied with cache instead of disks, the Read performance metrics are misleading, and you cannot generate consistent, repeatable results.
- An invocation of a single-instance test of either of these tools, one Write and Read test, shows how a single SAS job might perform at that time on the file system. If your system is already very busy with other users' workloads, this action might be adequate.
- A multiple-instance invocation of the tests shows what a concurrent SAS working load might experience, and is recommended for systems that are experiencing no other current jobs, or that have a very light workload.

- Conduct several of these test sets at various times of the day, and over several busy days to get a comprehensive profile of I/O characteristics on systems that run a variety of workloads.
- Perform a test during the same time of day and on the same file system or systems.
- Specify a file that you want to create in the file system that you want to test. If you are using batch mode, you can use the redirection operator (>) to send the output to a different file system. Here is an example: a .bat file with sasiotest.exe [...arguments...] > test.txt.
- The output from this tool is the megabytes written per second and megabytes read per second. Use these metrics to help analyze and tune I/O for file systems that support SAS.

Acceptable SAS I/O Performance

The test output from running the Sasiotest utility shows throughput results in MB per second. A minimum of 150MB per second per core is recommended for SAS file systems. See System Requirements.

Dictionary

SASIOTEST Command

Writes and reads files that are similar to SAS files in order to measure file I/O throughput on Windows.

Requirements: When you use the -sgio option, -pagesize must be a multiple of 4K on x86 systems and a multiple of 8K on x64 and IPF systems.

If the -sgio option is present, then -numpages is required.

The *filename* argument must be the first argument.

Syntax

SASIOTEST *filename <options>*

Required Argument

filename
specifies the filename that is read or written and can include the file system path.

Optional Argument

<options>

-w or -r
-w creates and writes the specified file.

-r reads the specified file.

Default -w

-seq or -ran
-seq reads sequentially from the specified file.

-ran reads randomly from the specified file.

Default -seq

-filesize *n*
specifies the size of the file to write or read. For example, *n* can be 800K, 512K, 5MB, 2GB.

Default If -filesize is not specified, the default is 1MB.

-pagesize <*n*>
specifies the size of the page for the I/O operation. For example, *n* can be 8K, 64K, 128K.

Default If -pagesize is not specified, the default is 8K.

-sgio
specifies whether Scatter Gather (SGIO) Read and SGIO Write APIs are used for I/O operations. The following options are specific to SGIO:

-numpages <*n*>
specifies the number of pages to read or write for each SGIO operation.

-unbuffered
specifies that the written pages bypass the file cache.

-writethru
specifies to write through any intermediate cache and go directly to disk.

Examples

Example 1

```
c:\m904_win>sasiotest test.out -w
SAS I/O Test Utility: v1.0

WARNING: Filesize option not specified. Defaulting to 1 MB filesize.
WARNING: Pagesize option not specified. Defaulting to 8K pagesize.
Sequentially writing 1048576 bytes to file: test.out with a 8192 pagesize.
Write Throughput: 71.100308 MB/Sec.
```

Example 2

```
c:\m904_win>sasiotest test.out -w -pagesize 64k
SAS I/O Test Utility: v1.0

WARNING: Filesize option not specified. Defaulting to 1 MB filesize.
Sequentially writing 1048576 bytes to file: test.out with a 65536 pagesize.
Write Throughput: 72.561777 MB/Sec.
```

Example 3

```
c:\m904_win>sasiotest test.out -w -pagesize 4k -filesize 2M
SAS I/O Test Utility: v1.0

Sequentially writing 2097152 bytes to file: test.out with a 4096 pagesize.
Write Throughput: 45.795315 MB/Sec.
```

Example 4

```
c:\m904_win>sasiotest test.out -r
SAS I/O Test Utility: v1.0

WARNING: Filesize option not specified. Defaulting to 1 MB filesize.
WARNING: Pagesize option not specified. Defaulting to 8K pagesize.
Sequentially reading 1048576 bytes from file: test.out with a 8192 pagesize.
Read Throughput: 258.301703 MB/Sec.
```

Note: Even though you are reading the file you just wrote, if you do not specify a PAGESIZE and FILESIZE argument, the sasiotest utility uses its default values of

8K and 1MB. See Example 5 for the correct way to read the file that was created in Example 3.

Example 5

```
c:\m904_win>sasiotest test.out -r -pagesize 4k -filesize 2M -numpages 2
SAS I/O Test Utility: v1.0
```

```
Sequentially reading 2097152 bytes from file: test.out with a 4096 pagesize.
Read Throughput: 237.732948 MB/Sec.
```

Example 6

```
c:\m904_win>sasiotest test.out -w -sgio -pagesize 4k -filesize 2G -
numpages 256
SAS I/O Test Utility: v1.0
```

```
Gather Writing 256 pages - 2 iterations to file: test.out with a 4096 pagesize.
Write Throughput: 18.363657 MB/Sec.
```

Example 7

```
c:\m904_win>sasiotest test.out -r -sgio -pagesize 4k -filesize 2G -
numpages 256
SAS I/O Test Utility: v1.0
```

```
Scatter Reading 256 pages - 2 iterations from file: test.out with a 4096 pagesize.
Read Throughput: 33.205119 MB/Sec.
```