Paper TU07

SAS® System Options are Your Friends

Edward Heaton, Westat, Rockville, MD

Abstract

Does *SAS*® always do things the way you want? Have you ever made a simple little mistake of omission with disastrous consequences? Do you find the *SAS* log difficult to read and less informative than you need? Do you keep doing the same little tasks over and over because *SAS* doesn't remember what you want?

System Options can be your solution to some of these problems.

What! This is a beginning tutorial. You want me to mess with *System Options*? Isn't that a little like playing around with the *Windows Registry*?

Not at all; and it isn't hard to pick out some *System Options* to make your code easier to manage.

SAS provides a lot of *System Options*. There are about 258 in version 8.2 for *Windows*. With version 9, ten of them went away and 68 new options were introduced. Although their implementation can sometimes be confusing, these options should not be ignored. The author will explain a little about their usage in a *Windows* environment and will present a handful that will make your work easier and less prone to simple mistakes.

Introduction

SAS System Options control the way your *SAS* session works. They are specified in various ways including:

- *SAS* default;
- The configuration file;
- The command line;
- The autoexec file;
- The *Options Environment Window*;
- The **`Options`** statement; and
- The **`opLoad`** procedure.

Some options can only be specified when *SAS* initializes; others can be set or changed at any time during a *SAS* session.

You can check the settings of your *SAS System Options* with the **`options`** procedure.

```
Proc options ;
Run ;
```

You can narrow down the options you print by specifying the *option group*…

```
Proc options group=errorHandling ;
Run ;
```

or a single option…

```
Proc options option=fmtErr ;
Run ;
```

You can get more information with the **`define`** and **`value`** options. The **`define`** option writes the option's description, type, and group to the *SAS* log. The **`value`** option writes the option's value and scope to the *SAS* log.

```
Proc options option=fmtErr value define ;
Run ;
```

Many *System Options* can be saved in a *SAS* data set somewhat the same way that value-label formats can be saved using the **cntlOut=** option in the **format** procedure. Code

```
Proc optSave out=libRef.dsName ;
Run ;
```

to save your options to a *SAS* data set and code

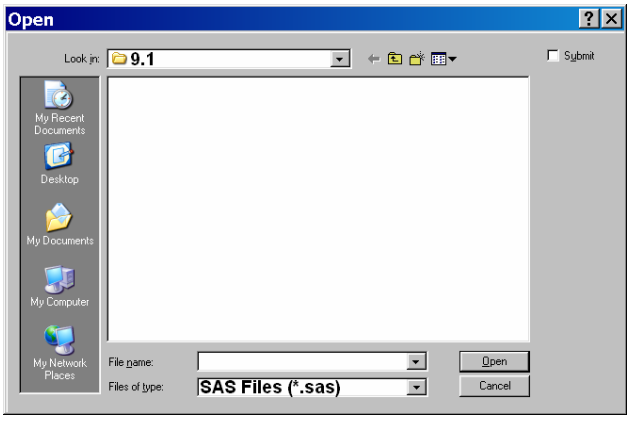
```
Proc optLoad data=libRef.dsName ;
Run ;
```

to set your options from the ***libRef.dsName*** data set.

Okay, let's look at some useful *System Options*.

1. SasInitialFolder="."

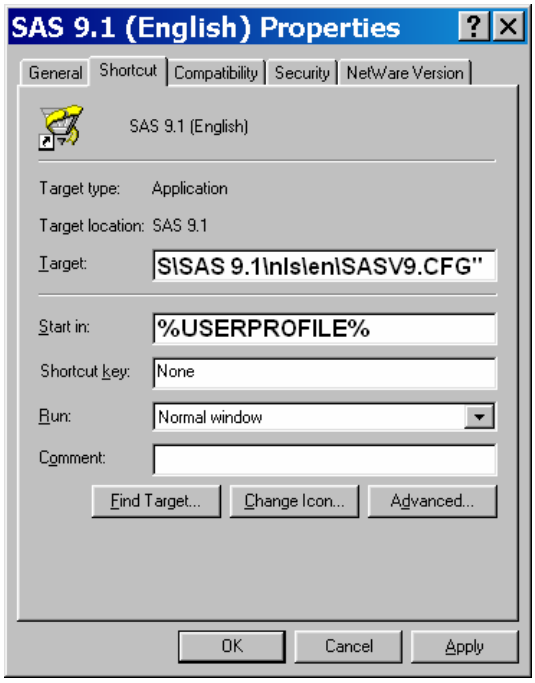
Do you work on projects that have a folder dedicated to that project? If you try to open a *SAS Program File* with the *Open* button or if you try to save your work with the *Save* button, you will get a dialog box that opens to a very *out-of-the-way* place, e.g., **C:\Documents and Settings\heaton_e\My Documents\My SAS Files\9.1**. (Your location will vary.).

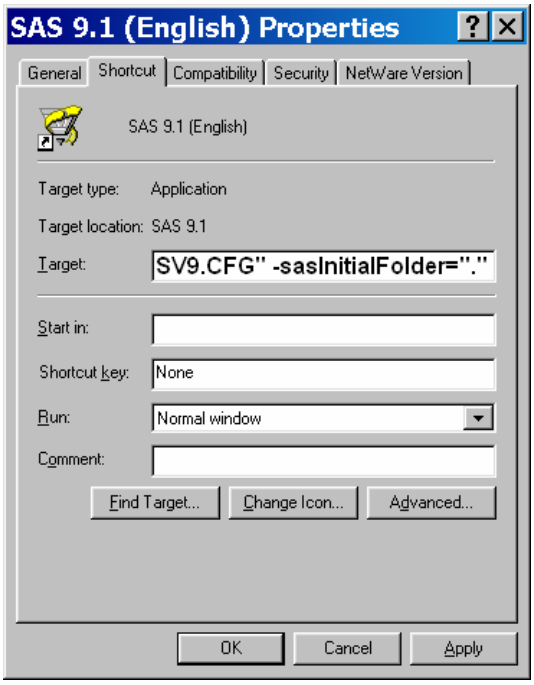


It's then an inconvenience to find the project folder before you can save or open your file.

You can make *SAS* use your project folder as the default folder with the **sasInitialFolder="."** *System Option*. Here is the process.

1. Copy the *SAS* icon from the **Start** menu to the project folder where you store your code for the project.
2. Alter the properties of that icon. (Right-click on the icon and select **Properties**.) We will need to change two things.
   a. Add **-sasInitialFolder="."** to the end to the **Target:** dialog box. This will tell *SAS* to show the ***Open*** and ***Save As*** windows at the location specified in the **Start in:** dialog box.
   b. Clear the **Start in:** dialog box. This will tell *Windows* to use the location of this shortcut as the **Start in:** location.





Now, we can open or save a file and *SAS* will start in the folder that contains this *SAS* icon. We can put copies of this shortcut in any folder that we want and *SAS* will start in that folder if we start *SAS* from the shortcut.

2. DkrOCond=noWarning

I often write code that has a **drop=** data set option something like in the following code.

```
Data RandomNumbers( drop=_: ) ;
    Format Key z3. ;
    Do _i=1 to 10 ;
        Key = ceil( ranUni(681732) * 100 ) ;
        Output ;
    End ;
Run ;
```

The **Do** loop counter is not wanted in my output data set so I started it with an underscore. I make all my temporary variable names start with an underscore. That way, I can drop all of them with the **drop=_:** data set option. In fact, I often add **(drop=_:)** as a data set option in my **Data** statements just in case I create temporary variables. (I never start the name of any permanent variable with an underscore.) The problem arises when I create no temporary variables in the **Data** step. Then I get a warning that there are no variables to drop. I don't like warnings in my log.

The solution: add

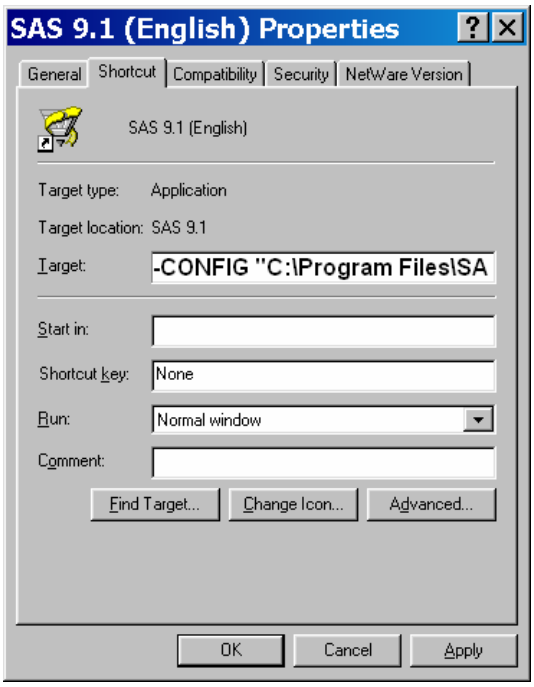
```
Options dkrOCond=noWarning ;
```

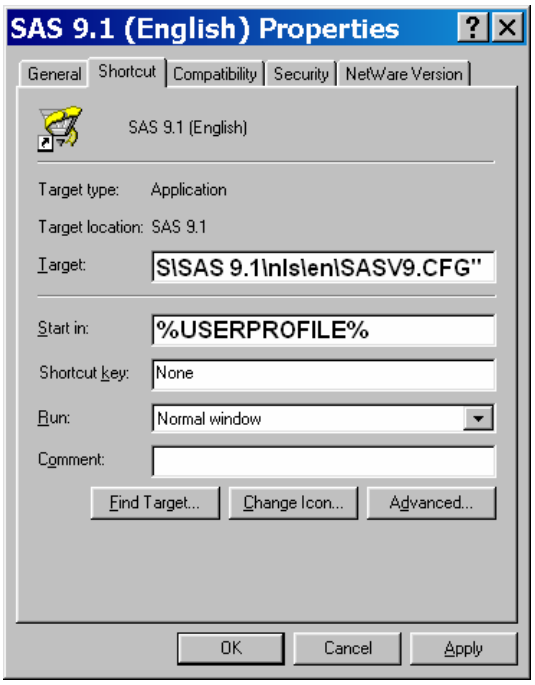
to ***autoExec.sas***. This ***drop-keep-rename output condition*** option controls the error detection for output data sets using the **drop=**, **keep=**, or **rename=** data set options. It also monitors the **Drop**, **Keep**, and **Rename** statements.

You might have to search for your copy of ***autoExec.sas***; my copy is in **C:\Program Files\SAS\SAS 9.1**. You may not even have such a file. Don't worry; just create one. It's an ASCII text file just like any other *SAS* program file. The secret is to put it in the right place; it should be in the same folder as your ***sas.exe***.

3. -EchoAuto

Since we have information in our ***autoExec.sas*** that is pertinent to the execution of our code, we might want that information to be part of our *SAS* log. We can accomplish this with the **`-echoAuto`** option. Of course, this *System Option* has to be available before *SAS* starts processing ***autoExec.sas***, so the option must be specified at *SAS* invocation. Let's add it to the *SAS* configuration file. You can find the location of that file by looking at the properties of your *SAS* icon.





You should be able to find, in the **Target:** dialog box, the **`-config`** option. In this example, the file is **`C:\Program Files\SAS\SAS 9.1\nls\en\SASV9.CFG`**. Find and open that file. It will contain lots of *SAS System Options*. You should be able to find a comment, something like…

```
/* DO NOT EDIT BELOW THIS LINE ... */
```

User options should be added above this comment. Find that text and add the following line of code just above the comment.

```
-echoAuto
```

Now, we get this message in our *SAS* log.

```
NOTE: AUTOEXEC processing beginning; file is
      C:\Program Files\SAS\ SAS 9.1\autoexec.sas.

1    Options dkrOCond=noWarning ;

NOTE: AUTOEXEC processing completed.
```

4. NoFmtErr

If you try to use a data set that has ***user-defined*** formats but haven't told *SAS* where to find those formats, the default setting of **`fmtErr`** will create an error and stop the process. You will get an error message in the log similar to the following for each user-defined format that could not be found.

```
ERROR: Format NUMS not found or couldn't be loaded for variable x.
```

If you change the setting to **noFmtErr**, then you can use the data set and the unformatted values will be printed. **NoFmtErr** replaces missing formats with the **w.** or **$w.** format, issues a note, and continues processing. Add this option to ***autoExec.sas***.

```
Options dkrOCond=noWarning noFmtErr ;
```

5. FmtSearch=( )

When *SAS* finds a reference to a user-defined *format* or *informat*, it will look for a format catalog called **work.Formats**. If it finds no such format catalog, or if it does not find the user-defined format in that catalog, it will look for a catalog called **library.Formats**. If it has no luck there, *SAS* will go no further unless you tell it where to look.

If you name a *format catalog* anything other than **Formats** or store it in a library with a *libRef* other than **library**, you will need to use the **fmtSearch=** option to tell *SAS* where to find the formats. List the format catalogs in *parentheses* starting with the one you want to search first. *SAS* searches the format catalogs in the order listed, until the desired member is found. The value of the catalog specification can be either the ***libRef*** or ***libRef.catalog***. If only the ***libRef*** is given, *SAS* assumes that **Formats** is the catalog name.

If they are not specified, the **work.Formats** catalog is searched first and the **library.Formats** catalog is searched next. If a catalog in the list does not exist, that particular item is ignored and searching continues – no problem.

6. MergeNoBy=error

Rarely do we perform a **Data** step merge without expecting *SAS* to merge on the values of certain variables. Sometimes, though, I ***inadvertently*** omit the **By** statement. That's a scary scenario for me; I don't want *SAS* to blindly do things that could get me in trouble. Fortunately, *SAS* allows us to tell it to refuse such an instruction with the **mergeNoBy=** *System Option*. The default is **noWarn**, but this is the last thing we should want. If we specify **mergeNoBy=warn** and code something like the following...

```
Options mergeNoBy=warn ;
Data AllOfMe ;
    Merge Part1 Part2 ;
Run ;
```

we will get a warning in the log.

```
WARNING: No BY statement was specified for a MERGE statement.
```

However, this is not a severe enough restriction. Change the option to **mergeNoBy=error** and *SAS* will stop the process.

```
ERROR: No BY statement was specified for a MERGE statement.

NOTE: The SAS System stopped processing this step because of
errors.

WARNING: The data set WORK.ALLOFME may be incomplete.  When this
         step was stopped there were 0 observations and 3
         variables.

WARNING: Data set WORK.ALLOFME was not replaced because this step
         was stopped.
```

This is much safer. Add this *System Option* to ***autoexec.sas***.

```
Options
    dkrOCond=noWarning
    noFmtErr
    mergeNoBy=error
;
```

On the ***rare*** occasion that you do want to merge with no **By** statement, code

```
Options mergeNoBy=noWarn ;
```

immediately before the **Data** step and

```
Options mergeNoBy=error ;
```

immediately after.

7. MsgLevel=i

Of course, our merges are still not as safe as we would like. If you ***match-merge*** two data sets and there are ***common*** variables other than the ones in the **By** statement, you have no way to know for sure which data set provided the values for these variables in the output data set. Consider the following two *SAS* data sets.

Head:

Key	x	h
6	Head	Head
6	Head	Head
22	Head	Head

Toe:

Key	x	t
6	Toe	Toe
22	Toe	Toe
22	Toe	Toe

Now, let's merge these two data sets by **key**.

```
Data HeadToToe ;
    Merge Head Toe ;
        By key ;
Run ;
```

Output:

Obs	key	x	h	t
1	6	Toe	Head	Toe
2	6	Head	Head	Toe
3	22	Toe	Head	Toe
4	22	Toe	Head	Toe

Of course, you do not want to perform such a merge because it would be very difficult to predict the source of each value of **x**. Nevertheless, by default, *SAS* gives no warning that you have this problem. Again, we want *SAS* to protect us from mistakes whenever it is easy for it to do so.

The **msgLevel=i** *System Option*, among other things, tells *SAS* to write a message to the *SAS* log whenever a **Merge** statement causes variables to be overwritten.

```
INFO: The variable x on data set WORK.HEAD will be overwritten by
      data set WORK.TOE.

NOTE: MERGE statement has more than one data set with repeats of
      BY values.
```

Now, the **`INFO:`** message is wrong. The value of the variable **`x`** from **`work.Toe`** was overwritten by data set **`work.Head`** to create observation 2. Still, I like having the message and I want the **`msgLevel=i`** *System Option* in my ***autoExec.sas***.

```
Options
    dkrOCond=noWarning
    noFmtErr
    mergeNoBy=error
    msgLevel=i
;
```

8. MPrint

I write ***macro*** code – lots of macro code. We can tell SAS to print the ***resolved*** code to the SAS log with the **`mPrint`** option. The log will then display SAS statements that are ***generated*** by macro execution. The statements are formatted with macro variable references and macro functions resolved, with each statement beginning on a new line and with one space between words. I like this, it helps with debugging and is a clear documentation of the process in the SAS log. We want this *System Option* to always be in effect, so put it in ***autoExec.sas***.

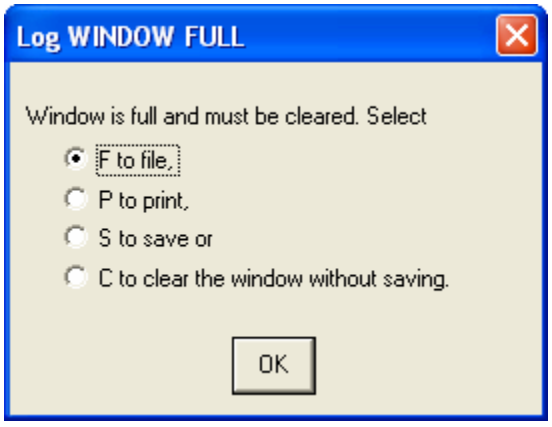
```
Options
    dkrOCond=noWarning
    noFmtErr
    mergeNoBy=error
    msgLevel=i
    mPrint
;
```

Version 9 Extras

Here are some new *System Options* that are available for *SAS 9.1.2*.

9. DmsLogSize=999999

Does your *SAS* log ever overflow with the following message.



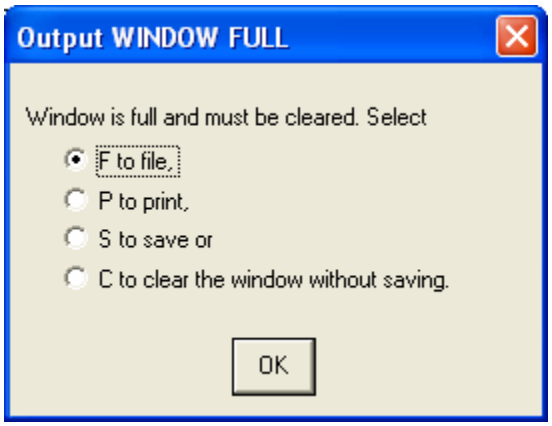
SAS, by default, only allows 99,999 lines in the log window. If you are getting this message for a job, code…

```
-dmsLogSize=999999
```

in your *SAS* configuration file or on your command line to keep *SAS* from stalling. Of course, you will still stall if you try to write a million lines to the *SAS* log!

10. DmsOutSize=999999

Similarly, your *SAS* session will stall if you try to write more than 99,999 lines to the *SAS Output* window.



If this is happening to you, code…

```
-dmsOutSize=999999
```

in your *SAS* configuration file or on your command line to keep *SAS* from stalling unless it is trying to write the one millionth line.

11. DtReset

SAS records the date and time at the start of every session uses that information in your output unless you have specified the **`noDate`** *System Option*. If you are running interactive *SAS* or a huge, time-consuming batch job, the date or time might not be appropriate. Your code can specify a date and time that reflects when a procedure starts to print a page with the **`dtReset`** *System Option*.

```
Options dtReset ;
```

Add this code for output that needs an accurate time stamp.

12. NoQuoteLenMax

While *SAS* will allow character-string constants to be as long as 32,767 characters, *SAS* will write a message to the log if you try to code a constant longer than 262 characters.

```
WARNING: The quoted string currently being processed has become
         more than 262 characters long.  You may have unbalanced
         quotation marks.
```

Without this message, you might find yourself searching back through more than ten pages of code to find your missing quotation mark!

Sometimes I use a macro variable to fill in the details between the quotation marks. Consider, for example, the following code, which creates a tab-delimited text file with the first record containing variable names.

```
Proc contents
    data=fileRef.DataSet
    out=Vars( keep= Name VarNum )
    noPrint
;
Run ;
Proc sql noPrint ;
    Select Name into :vars separated by " "
        from Vars
        order by VarNum
    ;
Quit ;
Data _null_ ;
    File out dlm="09"x lRecL=1024 ;
    If ( _n_ eq 1 ) then do ;
        Vars = translate("&vars" , "09"x , " " ) ;
        Put Vars ;
    End ;
    Set fileRef.DataSet ;
    Put &vars ;
Run ;
```

The list of variable names can easily exceed 262 characters. However, I don't want my customer to look at the log and get concerned with the warning. So, I will add…

```
Options noQuoteLenMax ;
```

immediately before the **Data** step to suppress the warning and then code…

```
Options quoteLenMax ;
```

after the end of the **Data** step.

13. ValidFmtName=

SAS 9.0 introduced long *format* and *informat* names. For some processes, this is a great benefit. However, we need to make assure that long names are not used whenever we have to deliver a *SAS* data set to someone who is using *SAS* version 8. Make sure all conversions work smoothly by using the **validFmtName=** *System Option*. This controls the ***length*** of the names of *formats* and *informats* that can be used when creating new *SAS* data sets. **ValidFmtName=long** specifies that *format* and *informat* names can be up to 32 characters. This is the default. **ValidFmtName=fail** specifies that using a *format* or *informat* name that is longer than eight characters results in an error message. If you explicitly specify the v7 or v8 engine (e.g., in a **LibName** statement), *SAS* automatically uses the **validFmtName=fail** behavior for data sets associated with those engines. **ValidFmtName=warn** is the same as **fail**, except that, if a *format* or *informat* name exceeds eight characters, *SAS* displays a warning message.

Disclaimer

The content of this paper is the work of the author and does not necessarily represent the opinions, recommendations, or practices of Westat.

Acknowledgments

I want to thank Ian Whitlock, Mike Rhoads, Michael Raithel, and Norma Neuberg for reviewing this paper and for their helpful suggestions.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Edward Heaton
Westat
1650 Research Boulevard
Rockville, MD 20850-3195
Phone: (301) 610-4818
Email: EdHeaton@Westat.com
URL: http://www.westat.com

Appendix 2

New System Options for SAS versions 9.0, 9.1, and 9.1.2.

Option	*Data Type*	*Description*	*Level*	*Group*
connectPersist	Boolean	Persist connection to remote session	portable	**communications**
signOnWait	Boolean	Wait for a SAS/CONNECT signon to finish before allowing further processing to occur	portable	**communications**
sslCertIss	char	Name of the certificate authority issuing digital certificate	host	**communications**
sslCertSerial	char	Serial number of the digital certificate	host	**communications**
sslCertSubj	char	Subject of the digital certificate	host	**communications**
sslClientAuth	Boolean	Require client authentication when a connection is established.	portable	**communications**
sslCrlCheck	Boolean	Check certificate revocation list when validating digital certificates	portable	**communications**
sysrPutSync	Boolean	Macro variables are set as soon as SYSRPUT executes.	portable	**communications**
dqLocale	char	Data Quality best locale default and search order	portable	**dataQuality**
dqSetupLoc	char	SAS Data Quality - Cleanse setup file	portable	**dataQuality**
emailAuthProtocol	char	Identifies the *SMTP* email authentication protocol	portable	**email**
autosaveLoc	char	Identifies the location where *program editor* contents are auto saved	portable	**envDisplay**
dmsLogSize	num	Maximum number of rows in *DMS* log window	portable	**envDisplay**
dmsOutSize	num	Maximum number of rows in *DMS* output window	portable	**envDisplay**
docIndex	char		portable	**envDisplay**
docTOC	char		portable	**envDisplay**
sleepWindow	Boolean	Disable idle window when SAS is waiting for the wakeup or sleep function to complete.	host	**envDisplay**
sysGuiFont	char	Sets the default *GUI* font	host	**envDisplay**
toolsMenu	Boolean	Allows the *Tools* pull-down to be displayed in the *SAS* menu	portable	**envDisplay**
viewMenu	Boolean	Allow the *View* pull-down to be displayed in the *SAS* menu	portable	**envDisplay**
sasHost	char		host	**envFiles**

Appendix 2

Option	*Data Type*	*Description*	*Level*	*Group*
uuIdCount	num	uuid *Generator Server Cache*	portable	**envFiles**
uuidGendHost	char	uuid *Generator Server*	portable	**envFiles**
dmsSynChk	Boolean	Enables syntax check, in windowing mode, for a submitted statement block	portable	**errorHandling**
errorByAbend	Boolean	Abend on **By**-group error condition	portable	**errorHandling**
quoteLenMax	Boolean	Enable warning for quoted string length max	portable	**errorHandling**
authProviderDomain	char	Authentication providers associated with domain suffixes	host	**execModes**
jreOptions	char	Java Runtime Environment options	host	**execModes**
syntaxCheck	Boolean	Puts *SAS* into syntax checking mode	portable	**execModes**
termStmt	char	Specifies *SAS* statements to execute at *SAS* termination	portable	**execModes**
helpEnCmd	Boolean	Use the English index to resolve command line help requests	portable	**help**
helpIndex	char	Location of *Help* index files	portable	**help**
helpTOC	char	Location of the *Help* table of contents files	portable	**help**
accessibility	char	Enable Extended Accessibility	host	**inputControl**
bySorted	Boolean	Requires *SAS* data set observations to be sorted for **By** processing	portable	**inputControl**
dateStyle	char	Identify sequence of month, day and year when ANYDATE informat data is ambiguous.	portable	**inputControl**
dtReset	Boolean	Updates date and time for log and print output	portable	**log_ListControl**
pageBreakInitial	Boolean	Begins *SAS* log and listing files on a new page.	portable	**log_ListControl**
logParm	char	Specifies *SAS* log file control parameters	portable	**logControl**
mAutoLocDisplay	Boolean	Display the location from which the autocall macro source code is compiled	portable	**macro**
mCompileNote	char	Issue a note to the log when a macro has been successfully compiled	portable	**macro**
mExecNote	Boolean	Display macro execution information	portable	**macro**
minDelimiter	char	Identifies the character to use as the delimiter of the macro IN operator	portable	**macro**

Appendix 2

Option	*Data Type*	*Description*	*Level*	*Group*
mLogicNest	Boolean	Display macro nesting information in MLOGIC output	portable	**macro**
mPrintNest	Boolean	Display macro nesting information in MPRINT output	portable	**macro**
memBlkSz	num	Size of memory blocks allocated to support MEMLIB and MEMCACHE options.	host	**memory**
memBlkSz	num	Size of memory blocks allocated to support **memLib** and **memCache** options.	host	**memory**
memMaxSz	num	Maximum amount of memory allocated to support **memLib** and **memCache** options.	host	**memory**
metaAutoResources	char	Startup automatic resource identifier	portable	**meta**
metaConnect	char	Identifies the named connection from the *MetaProfile* file to use as the default *Open Metadata Server* connection.	portable	**meta**
metaEncryptAlg	char	Specifies the type of encryption to use when communicating with an *Open Metadata Server*.	portable	**meta**
metaEncryptLevel	char	Specifies the encryption level to use when communicating with an *Open Metadata Server*.	portable	**meta**
metaId	char	Identifies the *SAS* installation to the *Metadata Server*	portable	**meta**
metaPass	char	Identifies the *Metadata Server* password	portable	**meta**
metaPort	num	Identifies the *Metadata Server* port	portable	**meta**
metaProfile	char	Specifies the location of the file containing meta profiles	portable	**meta**
metaProtocol	char	Identifies the *Metadata Server IOM* protocol	portable	**meta**
metaRepository	char	Identifies the *Metadata Server Repository* to use for storing *SAS* configuration information	portable	**meta**
metaServer	char	Identifies the *Metadata Server* where configuration information is saved and can be queried	portable	**meta**
metaUser	char	Identifies the *Metadata Server* userid	portable	**meta**
prtPersistDefault	Boolean	Persist the printer selected in *Print Setup* between *SAS* sessions.	host	**odsPrint**
textureLoc	char	Specifies the location of textures and images used by *ODS* styles.	portable	**odsPrint**
uPrintMenuSwitch	Boolean	Enable universal print menus when **universalPrint** is on.	host	**odsPrint**
armAgent	char	*ARM Agent* to use to collect *ARM* records	portable	**performance**

Appendix 2

Option	*Data Type*	*Description*	*Level*	*Group*
armLoc	char	Identifies location where *ARM* records are to be written	portable	**performance**
armSubsys	char	Enables or disables *ARM*ing of *SAS* subsystems	portable	**performance**
cpuCount	num	Number of processors available.	portable	**performance**
dbSliceParm	char	Alter DBMS engine threaded read behavior by expanding or disallowing threaded reads.	portable	**performance**
maxSegRatio	num	SPDE pre-evaluation phase time ratio	portable	**performance**
mExecSize	num	Maximum size for a macro to execute in memory	portable	**performance**
minPartSize	num	Minimum partition size when creating SPDE files	portable	**performance**
spdeIndexSortSize	num	Identifies memory to be used for SPDE asynchronous index create or append.	portable	**performance**
spdeMaxThreads	num	Maximum number of threads for SPDE processing	portable	**performance**
spdeSortSize	num	Memory for SPDE sort operations	portable	**performance**
spdeUtilLoc	char	Location where SPDE temporary utility files are created	portable	**performance**
spdeWhEval	char	Specifies SPDE WHERE statement evaluation strategy.	portable	**performance**
threads	Boolean	Threads are available for use with features of the *SAS System* that support threading	portable	**performance**
cmpLib	char	Identifies previously compiled libraries of *CMP* subroutines to use when linking	portable	**sasFiles**
iBufSize	num	Specifies the size of index file buffers	portable	**sasFiles**
utilLoc	char	Temporary utility file locations where procedures can create utility files	portable	**sasFiles**
v6CreateUpdate	char	Access to create and update *V6 SAS* files	portable	**sasFiles**
validFmtName	char	Controls the name length of informats and formats created and processed during a *SAS Session*	portable	**sasFiles**
sortCut	num	Specifies the number of observations above which the host sort program is used instead of the SAS sort program.	host	**sort**
sortCutP	num	Specifies the number of bytes above which the host sort program should be used instead of the SAS sort program.	host	**sort**
sortEquals	Boolean	Maintains the order for the input data set in the output data set, when processing identical **By**-variable values with **Proc sort**	portable	**sort**

Appendix 2