Paper 67-28

SAS® Systems 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?

No. 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 v8.2 for Windows. With version 9, 10 of them went away and we have 68 new options. Their implementation can be confusing. However, they 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 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 the 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 ;
```

Most of the *System Options* that can be reset in a SAS session can be saved in a SAS data set somewhat the same way that 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.

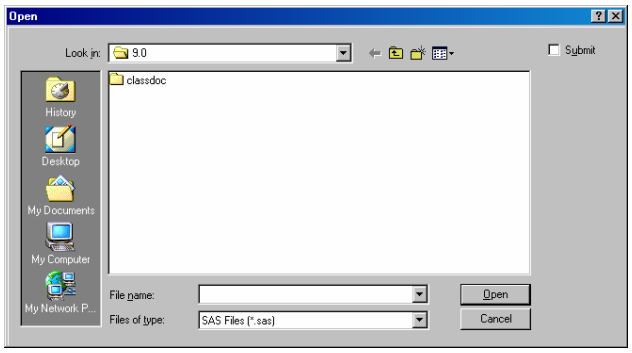
Okay, let's look at some useful *System Options*.

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:\Docum~1\EdHeaton\MyDocu~1\MySASF~1\V9
```

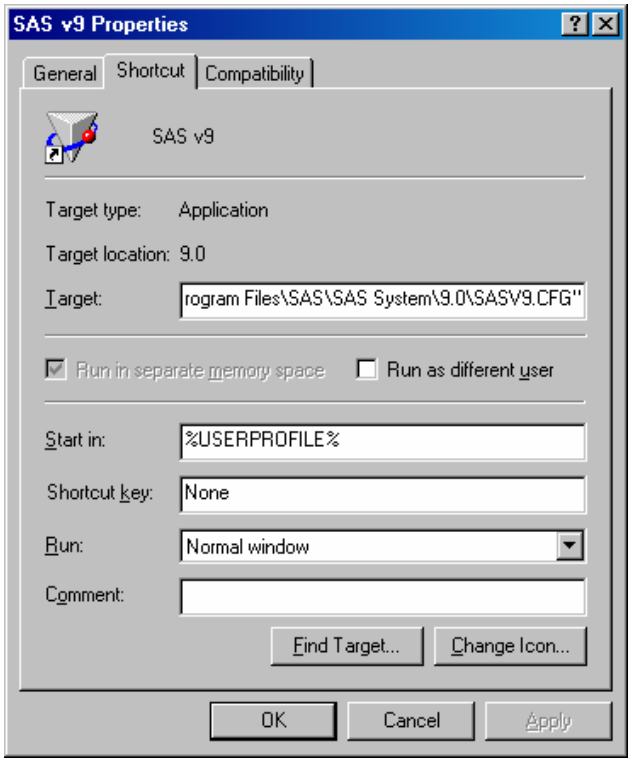
(Your location will vary.).



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.

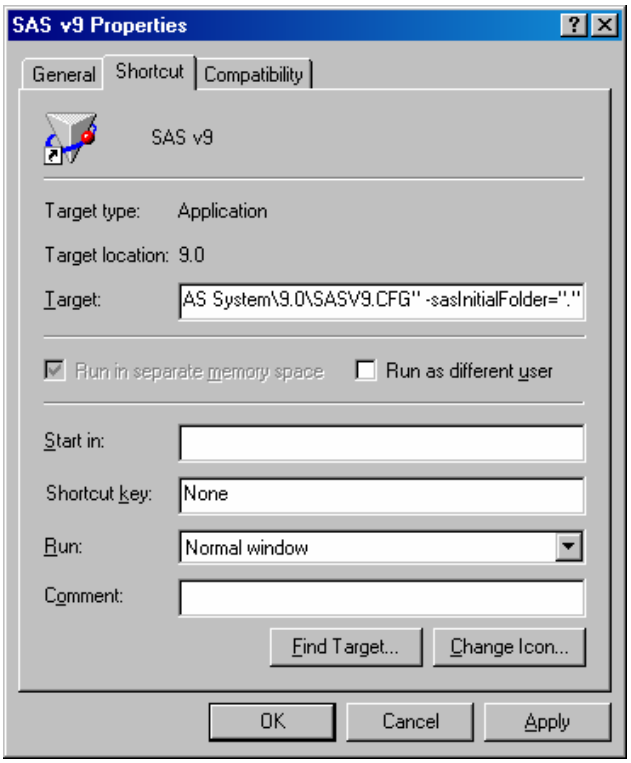


2. Alter the properties of that icon. (Right-click on the icon and select **Properties**.)

We will need to do 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.

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 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 that I might 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 were 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.

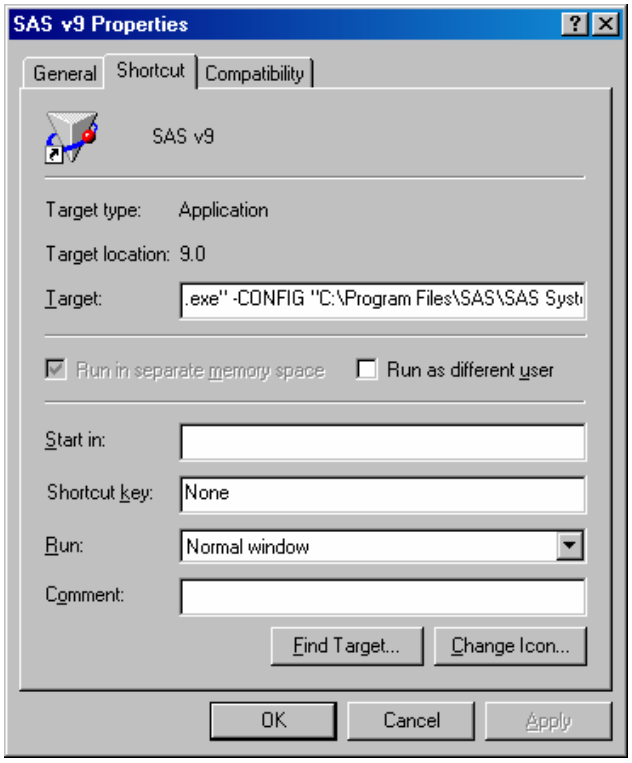
You might have to search for your copy of ***autoExec.sas***; my copy is in

```
C:\Program Files\SAS\SAS System\9.0
```

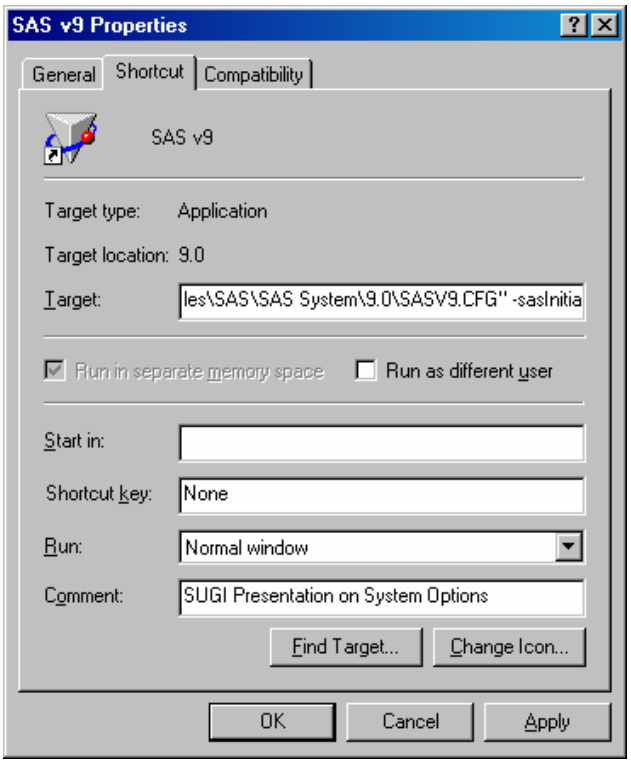
You may not even have such a file. Don't worry; just create one. It's an ASCII text file just like any other SAS program file. The secret is to put it in the right place; it should be in the same folder as your ***sas.exe***.

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 System\9.0\SASV9.CFG`.



Find and open that file. It will contain lots of SAS *System Options*. You should be able to find a comment line in the file that says something like

```
/* DO NOT EDIT BELOW THIS LINE - ┬ */
```

User options should be added above this comment. Find that line 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 System\9.0\
      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 ***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. I do, sometimes, ***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 assurance as to 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**. However, 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. However, 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 the *System Option* to always be in effect, so put it in ***autoExec.sas***.

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

Version 9 Extra

SAS will introduce long ***format*** and ***informat*** names in version 9. For some processes, this is a great benefit. However, we need to assure that those 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 alphanumeric 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, a ***warning*** message is displayed.

Conclusion

I have presented seven *System Options*; some were easy to implement, others were more difficult. Some were added to autoExec.sas, some to the configuration file, one to the start-up command line, and some were used in SAS Program files.

There are many SAS *System Options*. Take time to review these options; only you know which ones will help with your tasks. Then test the options to make sure you understand how they work and whether they really help.

With the proper application of SAS Systems Options, you can make your work more productive and less prone to errors. Embrace them, they are your friends.

Disclaimer

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

References

Acknowledgments

I want to thank Ian Whitlock, Mike Rhoads, and Michael Raithel 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
Work Phone: (301) 610-4818
Email: EdHeaton@Westat.com