

Paper 267-30

A SASautos Companion: Reusing Macros

Ronald Fehd, SAS-L's macro maven

Centers for Disease Control, and Prevention, Atlanta GA USA

Contents

Introduction	1
SASautos: the Environment Variable	1
Find the Configuration File:	1
Allocation:	2
Changing SASAUTOS :	2
SASautos: The Filename	2
Prepending to SASautos:	2
SASautos: The Option	3
Where SAS searches for macros	3
Macro Reuse	4
Example macros:	4
SAS knows nothing	4
Via %Include:	5
Autocall:	5
Compile And Store:	6
Masking:	9
Autoexec Examples	9
A Corporate Autoexec:	9
AutoexecSite:	10
Summary	10
Conclusion	11
Suggested Reading	11
Acknowledgements	11
Contact Information	11
Program ListMcat.sas	12

ABSTRACT

Reusable SAS® programs come in two forms: macros or %includes. This paper reviews the standard SAS environment and examines the options used to customize a session or batch program for reuse of macros stored in the project directory.

Expected audience is intermediate to advanced programmers and macro users.

Keywords: autocall compile macro masking MautoSource mStored program reuse SASautos SASmStore source store.

INTRODUCTION

Macros are available from several sources: SAS supplies many macros, users create their own and users may get macros from others. These macros are stored in several places: SAS stores its macros in different folders for each product licensed, user-created macros may be in programs, project folders, or site folders. This paper focuses on setting up a project to reuse macros stored in files in a project folder.

SASAUTOS is the keyword that we need to examine in order to make sense of the issues of storage and reuse of macros. SASAUTOS is used in three contexts, which I will list in the order of their loading during start up of a SAS session: the *environment variable* SASAUTOS which is allocated in your configuration file `SASv?.CFG`; the *filename* SASAUTOS lists the SAS-supplied folders containing macros; and the *option* SASAUTOS names the fileref(s) that SAS searches for macros.

Which SASautos?

1. the environment variable?
2. the filename?
3. the option?

SASAUTOS: THE ENVIRONMENT VARIABLE

The first usage of the term SASAUTOS is as one of the many environmental variables which are allocated — named, and assigned a value — in SAS's primary configuration file.

Find the Configuration File: The configuration file — `SASv?.cfg` — is located in SAS-home, the directory containing `SAS.exe`. I used the `-verbose` option on the command line to produce a log which has the file specifications of the configuration file(s).

```

_____ verbose.bat _____
1  "C:\Program Files\SAS Institute\SAS\V8\sas" -sysin verbose -log verbose8.log -verbose
2  "C:\Program Files\SAS\SAS 9.1\sas"      -sysin verbose -log verbose9.log -verbose

```

These logs show the names of the SAS-home directory and the name of each version's configuration file.

```

_____ verbose8.log _____
15  ==== Processed Configuration File(s) ====
16  C:\Program Files\SAS Institute\SAS\V8\sasv8.cfg

```

Note that v9 shows all — in this case, both — configuration files used during start up.

```

_____ verbose9.log _____
15  ==== Processed Configuration File(s) ====
16  C:\Program Files\SAS\SAS 9.1\sasv9.cfg
17  C:\Program Files\SAS\SAS 9.1\nls\en\SASV9.CFG

```

Allocation: Find and open your configuration file; search for
`-SET SASAUTOS`
 You should see the following in either of v8 or v9:

```

1  /* Setup the SAS System autocall library definition */
2  -SET SASAUTOS (
3      "!sasroot\core\sasmacro"
4      ... [deleted for brevity]
5      "!sasroot\stat\sasmacro"
6  )

```

Note that each directory specification (directory-spec) contains a reference to the environment variable SASroot: `!sasroot`. Find the `-SET SASROOT` allocation.

The value of the environment variable SASAUTOS is a list of directory specifications, each of which is enclosed in double quotation marks, delimited by space(s) and the set is surrounded by open and close parentheses. This is an important item to remember, which I will refer to later in the paragraph discussing prepending.

This log shows the use of the `sysget` function to show the value of environment variable SASAUTOS.

Note the opening parenthesis on line 4 and closing parenthesis on line 6.

```

1  %Put SASautos - Environment Variable;;
2  SASautos - Environment Variable:
3  %Put %sysget(SASautos);
4  (      "!sasroot\core\sasmacro"
5  ... [deleted for brevity]
6  "!sasroot\stat\sasmacro"      )

```

Changing SASAUTOS : You may edit your configuration file and add another directory specification in the space provided before the SAS-supplied items:

```

1  /* Setup the SAS System autocall library definition */
2  -SET SASAUTOS ("<My Directory with My SASmacros>"
3      "!sasroot\core\sasmacro"
4      ... [deleted for brevity]
5      "!sasroot\stat\sasmacro"
6  )

```

Is this a good idea? ... No! What is the first item that goes missing when you upgrade? Your customized configuration file! Where would be a good place to modify SASautos? How about in filename SASautos?

SASAUTOS: THE FILENAME

The filename SASAUTOS uses the environment variable SASAUTOS as its list of directory specifications. We can view the directories in filename SASautos with the `list` option: SAS-log line 1.
 ... Oop! ...

```

1  filename SASautos list;
2  WARNING: No logical assign for filename SASAUTOS.
3  %Include SASautos(af)/nosource2;
4  28      filename SASautos list;
5  NOTE: Fileref= SASAUTOS
6      Physical Name= C:\Program Files\SAS\SAS 9.1\core\sasmacro
7      ... [deleted for brevity]
8      Physical Name= C:\Program Files\SAS\SAS 9.1\stat\sasmacro

```

The filename is allocated when a macro reference is made. I include the first member of the core sasmacro directory — SAS-log line 2:28 — which causes SAS to allocate filename SASAUTOS and then the list option shows the directories.

Prepending to SASautos:
 The syntax for the filename statement is:

```

1  filename FileRef <directory specification> ;
2  or
3  filename FileRef ( <dir-spec-1 ... dir-spec-N> );

```

If you have macros stored in your current directory and you want to prepend this directory to — place it in front of — the SAS-supplied list of directories containing macros you can use this example code in your autoexec:

Note carefully that there is no close parenthesis for the list after line 2. The log shows this is a correct assignment: no warning or error after SAS-log line 3. The function `sysget` supplies both an open parenthesis — which is ignored — and a close parenthesis, which is used to close the list of directory specifications.

```

1      1      filename SASautos ("." %*current directory;
2      2      %sysget(SASautos)
3      3      ;%*closure of filename statement;
4      4      filename SASautos list;
5      NOTE: Fileref= SASAUTOS
6      Physical Name= L:\LaTeX\SUGI30
7      Physical Name= C:\Program Files\SAS\SAS 9.1\core\sasmacro
8      ... [deleted for brevity]
9      Physical Name= C:\Program Files\SAS\SAS 9.1\stat\sasmacro

```

SASAUTOS: THE OPTION

The option `SASAUTOS` is one or more file references (file-refs) which SAS searches for files with the name of a macro that has been called. The default file-ref is `SASAUTOS`, as shown here.

```

1      %Put SASautos<%sysfunc(getoption(SASautos))>;
2      PROC Options define value option = SASautos;

```

```

20     1      %Put SASautos<%sysfunc(getoption(SASautos))>;
21     SASautos<SASAUTOS>

```

```

33     Group= MACRO
34     Group Description: SAS macro language settings
35     Description: Search list for autocall macros

```

This concludes the review of the three contexts of `SASAUTOS`: environment variable, filename and option.

WHERE SAS SEARCHES FOR MACROS

SAS uses an algorithm of three steps when it encounters a macro call. First it searches work `SASmacr` catalog. This catalog contains macros which were used earlier in the session and have been compiled. Next it checks whether the compile and store option `SASmStore` is true; if so, then the user must have specified a libref for the option `SASmStore` and SAS will search option `SASmStore` `SASmacr` catalog for compiled and stored macros. Note-1: compile and store macros must have option `store`.

See the example program `TestS.sas`.

Note-2: v9 allows storage of source code for the macro.

See an example in `TestSv9.sas`.

Finally SAS checks whether the autocall option `MautoSource` is true; if so, then it searches for filenames of the macro in the option `SASautos` fileref which default is `SASautos`: the list of directories of SAS-supplied macros contained in the environment variable `SASautos` allocated in SAS configuration file. The file is `%included`, compiled and stored in the appropriate catalog.

SAS macro search algorithm		
	boolean	search
	true: always	work.SASmacr.catalog (session compiled macros)
if	option.Mstored	compiled and stored catalog default libref: none user-supplied libref: option.SASmStore.libref.SASmacr.catalog
if	option.MautoSource	autocall library: directory(s) default: option.SASautos.directory: fileref: SASautos user-modified: option.SASautos.directory-1 ... option.SASautos.directory-N when filename eq macroname found: %include filename.sas compile and store in appropriate catalog: work.SASmacr.catalog or option.SASmStore.libref.SASmacr.catalog

Note: The [SAS Online Doc] provides another description of the above process. See the paragraph: *Calling a Stored Compiled Macro* under the topic: *Saving Macros Using the Stored Compiled Macro Facility*.

In the next sections we will review methods of enabling SAS to use macros within a program that are stored in project directories.

MACRO REUSE

Reusable macros are stored in files with the macro name being also the filename. In most operating systems SAS will find the name whether the name of the file is of any combination of cases: lower case: `testx.sas`, mixed case: `TestX.sas` or upper case: `TESTX.SAS`. However, beware: in most versions of Unix, filename and macro name must be only lower case.

Example macros: TestX is used in the following demo. Macros TestY and TestZ differ from TestX only in name: they all have the same function: to write a note to the log.

```

_____ TestX.sas _____
1 %Macro TestX/des='TestX in file TestX'
2 ;%Put @@&SysMacroName. in file TestX.sas;%Mend;

```

SAS knows nothing of your external files.

```

_____ TestInclude.log _____
24 5 *macro TestX.sas is in a file;
25 6 *but calling the macro now generates an error;;
26 7
27 8 *open comment remove to execute macro :
28 9 %TestX
29 10 *which will generate this Warning and Error:
30 11 *WARNING: Apparent invocation of macro TESTX
31 12 not resolved.
32 13 *ERROR 180-322: Statement is not valid
33 14 * or it is used out of proper order.;
34 15
35 16 run cancel;*to recover and proceed with your demo;

```

Via %Include: A common method of bringing in macro definitions is to %include the file(s) into the program.

There are two styles of doing this: refer to the full name of the file, i.e., filename and extension as in SAS-log lines 20

and 26;

or provide a directory-specification (directory-spec) and refer only to the name of the file, SAS-log lines 32:33, in which case SAS expects the extension to be .sas.

Whichever case is used, the macros are available only in the job or session since they are compiled and stored in the work SASmacr catalog, which disappears at endSAS.

Note: A listing of program ListMcat is available after the contact information block.

```

37 18          *method 1:  include '<filespec>';
38 19          *method 1.1: filename.extension;
39 20          %Include      'TestX.sas';
40 23          %TestX
41 @@TESTX in file TestX.sas
42 24
43 25          *method 1.2: directory\filename.extension;
44 26          %Include 'L:\LaTeX\SUGI30\TestY.sas';
45 29          %TestY
46 @@TESTY in file TestY.sas
47 30
48 31          *method 2: include <directory-spec>(filename);
49 32          Filename Pgm '.';
50 33          %Include Pgm(TestZ);
51 36          %TestZ
52 @@TESTZ in file TestZ.sas
53 37
54 38          *macros are compiled and stored;
55 39          %Include Pgm(ListMcat);
56 ListMcat: List Macros in Catalog(s)
57 libname objname objdesc
58 WORK      TESTX      TestX in file TestX
59 WORK      TESTY      TestY in file TestY
60 WORK      TESTZ      TestZ in file TestZ

```

Including files containing macros in each program before they are used is not considered an optimal method of bringing macros into programs for several reasons: If option source2 is on, then the log will be filled with the macro statements. If method 1.2: full file specification, is used then if the file is moved the program breaks. This practice is known as hard-coding: file must be in that exact location. If the file is copied to a new location and updated, the program runs with an old version. When programs are consolidated into larger programs sub-programs including macros may duplicate the inclusion of other sub-programs.

Here's my favorite SAS-L signature on the badness of repetition:

Repetition obfuscates!
 Repetition reduction enhances elegance!
 Repetition reduction furthers finesse!

Autocall: A second method of making macros available is to have SAS search for them. Option MautoSource turns on the autocall facility for SAS-supplied macros in the fileref SASAUTOS, the single fileref of option sasautos. However, SAS does not know where to look for user-written macros.

Note: Macro Verify is in directory SAS ...core sasmacro. See the full path is section: SASautos The Filename, program ViewSASautosFileName9.log.

```

24 5          filename Pgm '.';
25 6          options MautoSource;*for SAS-supplied macros;
26 7          %Put position of I is %verify(TIME,TEMP).;
27 position of I is 2.
28 8          %Include Pgm(ListMcat);
29 ListMcat: List Macros in Catalog(s)
30 libname objname objdesc
31 WORK      VERIFY
32 86
33 87          *open comment remove to call macro
34 88          %TestX
35 89          *which generates this Warning and Error:
36 90          *WARNING: Apparent invocation of macro TESTX
37 91          *      not resolved.
38 92          *ERROR 180-322: Statement is not valid
39 93          *      or it is used out of proper order.;
40 94          run cancel;*note: unable to recover;

```

Option MautoSource is paired with option sasautos which must contain an expanded search path for project or site macros. Upon first usage, the file with the name of the macro is found in the filerefs in the option sasautos is %included, compiled and stored in the work SASmacr catalog and then executed.

```

24 _____ TestAutoCallUser.log _____
25 5      Filename          Pgm '.';
26 6      options  SASautos = (Pgm SASautos) MautoSource;
27 7
28 8      *SAS finds TestX.sas in directory Pgm;
29 9      *   includes compiles and stores
30 10     *   in Work.SASmacr.catalog;
31 11     %TestX
32 @@TESTX in file TestX.sas
33 12     %Include Pgm(ListMcat);
34 ListMcat: List Macros in Catalog(s)
35 libname objname objdesc
WORK      TESTX      TestX in file TestX

```

Compile And Store: A third method is to %include, compile and store the macros in a catalog other than work SASmacr catalog.

Macro TestS is used in the following demo.

Macro TestSv9 illustrates the use of the v9 option source.

```

_____ TestS.sas _____
1 %Macro TestS/des='TestS stored in file TestS' store
2 ;%Put @@&SysMacroName. stored in file TestS.sas;%Mend;

```

```

_____ TestSv9.sas _____
1 %Macro TestSv9/des = 'TestS stored in file TestS'
2   store source /*source is a v9 option*/
3 ;%Put @@&SysMacroName. stored in file TestSv9.sas;%Mend;

```

This log shows the errors associated with incorrect usage of the compile and store options: Mstored and SASMStore.

```

_____ TestCompileNStoreErr.log _____
24 5      options noxwait;
25 6      * zap Library.SASmacr;
26 7      x del          SASmacr.sas7bcat
27 7      !
28 8      Filename          Pgm '.';
29 9      options  SASautos = (Pgm SASautos) MautoSource;
30 10
31 11     *note: open comment remove to call macro:
32 12     %TestS
33 13     *which generates these Errors and Note:
34 14     *ERROR: The MSTORED option must be set
35 15     *   to use the /STORE macro statement option.
36 16     *ERROR: A dummy macro will be compiled.
37 17     *NOTE: Autocall member, TESTS,
38 18     *   has not been compiled;
39 19     options Mstored;
40 20
41 21     *note: open comment remove to call macro:
42 22     %TestS
43 23     *which generates these Errors
44 24     *ERROR: The option SASMSTORE = libref is not set.
45 25     *ERROR: A dummy macro will be compiled.;

```

In SAS-log line 9: the libref of the catalog which will contain compiled and stored macros must be named in the option SASmStore, SAS-log line 11.

Note: SAS-log line 16:18. I think that a more correct message would say that the macro has been compiled, it was not stored in work SAS-macr catalog but it was stored in the libref named in SASmStore.

Note: the log reports macro TestS has been stored in library SASmacr catalog: listing line 54, after SAS-log line 28.

Option SASmStore accepts only one libref. This limitation can be worked around by assigning two catalogs in one libref. See [Carpenter-2004, ch 12].

This program shows how to use the SAS-supplied macro CompStor.

Listing line 8:

Option McompileNote writes a note to the log when compile is successful.

Option MautoLocDisplay displays the full file-specification of the autocall macro source code.

```

----- TestCompileNStoreOk.log -----
28      8      Filename      Pgm      '.';
29      9      Libname      SiteMlib '.';
30 NOTE: Libref SITEMLIB was successfully assigned as follows:
31      Engine:      V9
32      Physical Name: L:\LaTeX\SUGI30
33 10      options SASautos = (Pgm SASautos) MautoSource
34 11      SASmStore = SiteMlib Mstored;
35 12
36 13      *note: open comment remove to call macro:
37 14      %TestS
38 15      *which generates this Note
39 16      *NOTE: Autocall member, TESTZ,
40 17      *
41 18      *      has not been compiled
42 19      *      by the macro processor.
43 20      *      It may contain a macro syntax error
44 21      *      or not define a macro with the same name
45 22      *      as the member.
46 23      *      To autocall this member again,
47 24      *      set OPTION MRECALL.;
48 24      %Include Pgm(TestS);
49 27      %TestS
50 @@TESTS stored in file TestS.sas
51 28      %Include Pgm(ListMcat);
52 ListMcat: SASmStore=SiteMlib
53 ListMcat: List Macros in Catalog(s)
54 libname objname objdesc
SITEMLIB TESTS      TestS stored in file TestS

```

```

----- TestCompileNStoreX2.sas -----
5 Libname      StorProj '<dir-spec project>';
6 Libname      StorSite '<dir-spec site >';
7 Libname      LibMacro (StorProj StorSite) ;
8 options SASmStore = LibMacro Mstored;

```

```

----- CompStorSAS.sas -----
1 /* name: CompStorSAS.sas
2 purpose: compile and store SAS-supplied core macros
3 see ...\SAS\...\core\sasmacro\compstor.sas
4 see also CompileAndStore.sas
5 note: pathname is directory of
6      site-macro-lib.SASmacro.catalog
7 *** ..... */
8 options MautoSource McompileNote = all MautoLocDisplay;
9 %CompStor(pathname=.);

```

This program illustrates the steps necessary to compile and store into a particular directory.

Listing line 25:

Option McompileNote writes a note to the log when compile is successful.

Option MautoLocDisplay displays the full file-specification of the autocall macro source code.

To display the macro catalog use either of the following programs:

```

1  CompileAndStore.sas
2  /* name: CompileAndStore.sas
3  -----: User Requirements:
4  purpose: create catalog of compiled macros
5  description: %include macro source code, compile and store
6  -----: Program Specifications:
7  program group: maintenance of macro catalog
8  program type: routine
9  SAS type: job
10 input: macros in current directory:
11 !SASsite\macroLibrary
12 process: %Include each macro
13 output: SASmacr.catalog
14 log and list of compiled macros
15 notes: required: macros must have option store
16 option : macros may have option source: v9
17 usage: batch submit
18 author: Ronald.Fehd@cdc.hhs.gov
19 note: see CompStorSAS to load SAS-supplied core macros
20 change notes:
21 RJF2 05Jan31 for SUGI30 SASautos paper
22 /*..... */
23 filename Macro '.';
24 libname SiteMacr '.';
25 options SASmStore = SiteMacr mStored
26 McompileNote = all MautoLocDisplay;
27 %Include Macro(Array);
28 %Include Macro(Nobs);
29 *Include Macro(...);
30
31 /*use utility from paper to show SASmaro.catalog **
32 filename Pgm 'L:\LaTeX\SUGI30';*!SASsite\includes';
33 %Include Pgm(ListMcat)/nosource2;
34 %Include Pgm(ProcCatalogSASmStoreSASmacr)/nosource2;
35 %Include Pgm(ProcSQLSASmStoreSASmacr)/nosource2;
36 /*****/

```

```

1  ProcCatalogSASmStoreSASmacr.sas
2  /* name: ProcCatalogSASmStoreSASmacr.sas
3  purpose: list compiled and stored macros
4  ***..... */
5  PROC Catalog
6  catalog = %sysfunc(getoption(SASmStore)).SASmacr;
7  contents;
8  quit;
9  run;

```

```

1  ProcSQLSASmStoreSASmacr.sas
2  /* name: ProcSQLSASmStoreSASmacr.sas
3  purpose: list compiled and stored macros
4  ***..... */
5  proc SQL;select *
6  from Dictionary.Catalogs
7  where MemName in ('SASMACR')
8  ;quit;
9  run;

```

Masking: Macro TestSinProgramA is used in the following demo. It contains a different version of macro TestS.

Masking occurs when a user-written macro has the same name as another, usually SAS-supplied, macro. In this demonstration case, macro TestS has been compiled and stored; another program, TestSinProgramA, contains a non-stored macro named TestS.

As shown on the previous page the stored version of TestS works correctly. However after program TestSinProgramA is executed, the definition of TestS has changed. This is a very hard problem to solve. I wrote the utility program ListMcat to help in identifying the occurrence of two macros with the same name in different catalogs. The listing appears after the contact information block.

AUTOEXEC EXAMPLES

Here is an example of the set of statements for an autoexec which enables the autoexec feature and compilation and storage of macros.

AutoCallDemo.log shows that the session starts — listing line 30:33 — with macros ready for access.

At SAS-log line 82, macro TestX is called: SAS searches, finds, compiles, stores and executes it.

A Corporate Autoexec: A Concerned Reader provided this example:

```

_____ TestSinProgramA.sas _____
1  %Macro TestS/des='TestS in file TestSinProgramA'
2  ;%Put @@&SysMacroName. in file TestSinProgramA.sas;%Mend;

```

```

_____ TestCompileNStoreOK.log continued _____
56  107      %Include Pgm(TestSinProgramA);
57  110      %TestS
58  @@TESTS in file TestSinProgramA.sas
59  111      %Include Pgm(ListMcat);
60  ListMcat: SASmStore=SiteMlib
61  ListMcat: List Macros in Catalog(s)
62  libname objname objdesc
63  SITEMLIB TESTS      TestS stored in file TestS
64  WORK      TESTS      TestS in file TestSinProgramA

```

```

_____ autoexecExample.sas _____
1  * name: autoexecExample.sas;
2  Filename      Pgm      '.';
3  Libname       MacroLib '.';
4  options      SASautos = (Pgm SASautos) MautoSource
5              SASmStore = MacroLib      Mstored;

```

```

_____ AutoCallDemo.log _____
29  2      %Include Pgm(ListMcat);
30  ListMcat: SASmStore=MacroLib
31  ListMcat: List Macros in Catalog(s)
32  libname objname objdesc
33  MACROLIB TESTS      TestS stored in file TestS
34  80      %TestS
35  @@TESTS stored in file TestS.sas
36  81
37  82      %TestX
38  @@TESTX in file TestX.sas
39  83      %Include Pgm(ListMcat);
40  ListMcat: SASmStore=MacroLib
41  ListMcat: List Macros in Catalog(s)
42  libname objname objdesc
43  MACROLIB TESTS      TestS stored in file TestS
44  WORK      TESTX      TestX in file TestX

```

```

_____ autoexecCorp.sas _____
1  *name: a Corporate autoexec.sas;
2
3  FileName SASmacro %sysGet(sasAutos) ;
4
5  Options  sasAutos=("!ProjMacs" /* project library, if any */
6              "L:\CorpMacro" /* corporate library */
7              SASmacro /* SAS macro libraries */
8              )
9              MautoSource;

```

AutoexecSite: The author uses this autoexec:

```

1  /*          autoexecSite.sas
2  /*          name: autoexecSite.sas
3  -----: User Requirements:
4  purpose: standardization of filename Pgm, SASautos
5  description: concatenate site folders
6                with SAS-supplied folders
7  -----: Program Specifications:
8  program group: invocation
9  program type: routine
10 SAS type: include w/parms from configuration
11 input: see project configuration file
12 process: allocation of filerefs, librefs, options
13 output: to log if invoked with -autoecho
14 notes: calls project autoexec
15 usage: -config u:\dls\SAS\sitev9?.cfg contains:
16         -SET          SITEroot "u:\dls\SAS"
17         -autoexec "!SITEroot\autoexecSite.sas"
18 author: Ronald.Fehd@cdc.hhs.gov
19 note: project libref:Library allocated in project autoexec
20 note: %sysget(SiteRoot) could be !SiteRoot
21 change notes:
22 RJF2 02Jul31 original
23 RJF2 05Feb02 polish and trim for SUGI30 SASautos paper
24 *** ..... */
25 %Put @@ ----- autoexecSite.sas -----;
26
27 *concatenate project directory with site includes;
28 Filename Pgm ('.' "%sysget(SiteRoot)\includes");
29
30 *prepend project and site macros to SAS-supplied macros;
31 Filename SASautos ('.' "%sysget(SiteRoot)\macros"
32                   "%sysget(SASautos)
33                   ;*correct: no close paren for filename;
34
35 Libname SiteMlib "!SITEroot\macroCat";
36
37 options dtReset          %*V9: date+time reset in title1;
38         MautoSource      %*autocall := on;
39         Mstored           %*compiled macros stored;
40         SASMstore = SiteMlib %*in SASmacr.catalog;
41
42 %Put Pgm(AutoExec) begin; %Inc Pgm(AutoExec);
43 %Put Pgm(AutoExec) end;
44 run;%Put @@ ----- autoexecSite.sas ----- end;

```

A Concerned Reader offers the opinion that using *here* — "." or — ". ." — as a directory specification in either of filename or libname is problematic for those on the help desk. *Here* depends on either the icon or shortcut value Start-In or the option SASinitialFolder. Sending both the program and its log would be helpful to help desk personnel who would then know exactly where the program was executed, and to be on the lookout for the associated configuration file and autoexec.

SUMMARY

There are a number of areas to pay careful attention to when setting up macro usage. Extreme care should be exercised when writing configuration and autoexec.sas files. This paper illustrates the use of *here* as an argument to filename and libname statements. This may be appropriate and adequate for research and development usage; it would probably not be appropriate for production usage. There are tradeoffs for the use of the macro option `store: compile and store` requires an extra step between development and production usage. While it is a time-saver and should be considered as an appropriate optimization technique, the problem of masking is a very subtle error and quite hard to comprehend in the SAS environment, so great care should be taken in choice of macro names in programs. [Carpenter-2004, ch 12] provides more information on building and using macro libraries.

CONCLUSION

SASAUTOS is a name used in three contexts: environmental variable, fileref of the filename statement, and an option which names filrefs which the autocall facility searches. Options associated with SASautos are the other autocall option: MautoSource, and the options of the compile and store facility: SASmStore and Mstore. Each can be modified to enable better use of SAS and system resources. The recommended place to set these options is in a project or site `autoexec.sas`.

REFERENCES

[SAS Online Doc] *SAS 9.1 Help and Documentation*, SAS Institute, Cary NC, 2004.

[Carpenter-2004] Carpenter, Art, 2004. *Art Carpenter's Complete Guide to the SAS[®] Macro Language, Second Edition* Cary NC: SAS Institute.

SUGGESTED READING

The program header information used in each of programs `CompileAndStore.sas`, `AutoExecSite.sas`, and `ListM-Cat.sas` is described in this paper:

Fehd, Ronald (2005), *Journeyman's Tools: The Writing for Reading and Reuse Program Header*, Proceedings of the 30th Annual SAS[®] Users Group International Conference, 2005.

ACKNOWLEDGEMENTS

The signature on repetition is the result of a conversation with Greg Nelson on the badness of inclusion of macros. Dianne Rhodes provided op-ed commentary, critique, and proofreading for which I am grateful. The whizards of SAS-L, in particular the European contingent, have contributed much to my knowledge of the intricacies of configuration and `autoexec` usage.

CONTACT INFORMATION

To receive the latest edition of any programs listed in this paper send an e-mail to the author with the subject:

`request SASautos programs`

Author: Ronald Fehd <mailto:Ronald.Fehd@cdc.hhs.gov>
Centers for Disease Control MS-G23 **e-mail: RJF2@cdc.gov**
4770 Buford Hwy NE
Atlanta GA 30341-3724 **bus: 770/488-8102**

about the author:

education:	B.S. Computer Science SUGI attendee since 1988 SAS-L reader since 1994
experience:	programmer: 20+ years data manager at CDC, using SAS: 17 years author: 10+ SUG papers
SAS-L:	author: 3,000+ messages to SAS-L since 1997 MVS: Most Valuable SAS-L contributor, 2001 HOF: Hall of Fame: MVS, second time, 2003

Document Production: This paper was typeset in \LaTeX . For further information about using \LaTeX to write your SUG paper, consult the SAS-L archives:

<http://www.listserv.uga.edu/cgi-bin/wa?S1=sas-l>
 Search for :
 The subject is or contains: LaTeX
 The author's address : RJF2
 Since : 01 June 2003

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.

PROGRAM ListMcat.sas

```

1      /*      name: ListMcat.sas
2      -----: User Requirements:
3          purpose: to show masked macros:
4              two same-named macros in different catalogs
5          description: List Macros in Catalogs
6              notes written to log
7                  of LibName, MacroName, Description
8      -----: Program Specifications:
9      program group: testing for masking: aka name collision
10     program type: routine
11     SAS type: include w/o parms
12     input: SAShelp.VCatalog
13     process: read from SAShelp.vCatalog
14             put vars
15     output: notes in log
16     notes: see also macro PutMasked
17             in book: A SAS Companion
18     usage: filename Pgm '.';
19             %Include Pgm(ListMcat);
20     author: Ronald.Fehd@cdc.hhs.gov
21     change notes:
22     RJF2 05Jan23 for SUGI30 SASautos paper
23     RJF2 05Jan30 compile&store MemName is always SASmacr
24     /*..... */
25     options nonotes;
26     %let WidthObjName = 32;%*max width macro name::ObjName;
27     %let WidthObjName = 7;%*for demo: length(objname) = 7;
28     DATA _Null_;
29     file LOG;%*ruler for column value assignments;
30     *   .   1   .   2   .   3   .   4   .   5
31     .  .5.  .0.  .5.  .0.  .5.  .0.  .5.  .0.  .5.  .01234
32     ListMcat: List Macros in Catalog(s)
33     libname8 objname8901234567890123456789012 objdesc
34     LIBRARY TESTY                                TestY
35     ;*end ruler;
36     retain SASmStore "%sysfunc(getoption(SASmstore))"
37             C1 1
38             C2 10
39             C3 44;
40             C3 = sum(C2,&WidthObjName.,2);
41     if SASmstore ne ' ' then
42     put 'ListMcat: ' SASmStore=;
43     put 'ListMcat: List Macros in Catalog(s)'
44         / @C1 'libname'
45           @C2 'objname'
46           @C3 'objdesc';
47     do until(EndoFile);
48     set SAShelp.VCATALG(where = (MemName eq 'SASMACR'))
49         end = EndoFile;
50     Put @C1 LibName
51         @C2 ObjName
52         @C3 ObjDesc; end; stop;
53     run;%SymDel WidthObjName;options notes;run;

```

This is the last page of the paper: A SASautos Companion: Reusing Macros.