Paper TU-05

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.

When viewing with
Adobe Acrobat
Press Here to
View: Fit Width

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:

SASv?.cfg: SASautos allocation

```
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.

ViewSASautosEnvVar9.log

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

SASv?.cfg: modifying SASautos allocation

```
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! . . .

ViewSASautosFilename9.log

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

filename syntax

```
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.

FileNameSASautosEx9.log

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

SASAUTOS: THE OPTION

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

ViewSASautosOption.sas

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

ViewSASautosOption9.log

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

ViewSASautosOption9.log

```
    Group= MACRO
    Group Description: SAS macro language settings
    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 earler 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;
```

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;
```

SAS knows nothing of your external files.

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.

TestInclude.log continued

```
18          *method 1:   include '<filespec>';
19          *method 1.1: filename.extension;
20          %Include       'TestX.sas';
23          %TestX
@@TESTX in file TestX.sas
24
25          *method 1.2: directory\filename.extension;
26          %Include 'L:\LaTeX\SUGI30\TestY.sas';
29          %TestY
@@TESTY in file TestY.sas
30
31          *method 2: include <directory-spec>(filename);
32          Filename Pgm '.';
33          %Include Pgm(TestZ);
36          %TestZ
@@TESTZ in file TestZ.sas
37
38          *macros are compiled and stored;
39          %Include Pgm(ListMcat);
ListMcat: List Macros in Catalog(s)
libname  objname  objdesc
WORK     TESTX    TestX in file TestX
WORK     TESTY    TestY in file TestY
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.

TestAutoCallSAS.log

```
5           filename Pgm '.';
6           options  MautoSource;*for SAS-supplied macros;
7           %Put position of I is %verify(TIME,TEMP).;
position of I is 2.
8           %Include Pgm(ListMcat);
ListMcat: List Macros in Catalog(s)
libname  objname  objdesc
WORK     VERIFY
86
87          *open comment remove to call macro
88          %TestX
89          *which generates this Warning and Error:
90          *WARNING: Apparent invocation of macro TESTX
91          *         not resolved.
92          *ERROR 180-322: Statement is not valid
93          *      or it is used out of proper order.;
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.

TestAutoCallUser.log

```
5          Filename              Pgm '.';
6          options  SASautos = (Pgm SASautos) MautoSource;
7
8          *SAS finds TestX.sas in directory Pgm;
9          *    includes compiles and stores
10         *    in Work.SASmacr.catalog;
11         %TestX
@@TESTX in file TestX.sas
12         %Include Pgm(ListMcat);
ListMcat: List Macros in Catalog(s)
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.

TestS.sas

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

Macro TestS is used in the following demo.

TestSv9.sas

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

Macro TestSv9 illustrates the use of the v9 option `source`.

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

TestCompileNStoreErr.log

```
5          options noxwait;
6          * zap Library.SASmacr;
7          x del          SASmacr.sas7bcat
7        !                                ;
8          Filename                 Pgm '.';
9          options  SASautos    = (Pgm SASautos) MautoSource;
10
11         *note: open comment remove to call macro:
12         %TestS
13         *which generates these Errors and Note:
14         *ERROR: The MSTORED option must be set
15         *       to use the /STORE macro statement option.
16         *ERROR: A dummy macro will be compiled.
17         *NOTE: Autocall member, TESTS,
18         *                       has not been compiled;
19         options Mstored;
20
21         *note: open comment remove to call macro:
22         %TestS
23         *which generates these Errors
24         *ERROR: The option SASMSTORE = libref is not set.
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 SASmacr 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.

TestCompileNStoreOk.log

```
8          Filename             Pgm      '.';
9          Libname              SiteMlib '.';
NOTE: Libref SITEMLIB was successfully assigned as follows:
      Engine:        V9
      Physical Name: L:\LaTeX\SUGI30
10         options  SASautos = (Pgm SASautos) MautoSource
11                 SASmStore =  SiteMlib      Mstored;
12
13         *note: open comment remove to call macro:
14         %TestS
15         *which generates this Note
16         *NOTE: Autocall member, TESTZ,
17         *                       has not been compiled
18         *      by the macro processor.
19         *      It may contain a macro syntax error
20         *      or not define a macro with the same name
21         *         as the member.
22         *      To autocall this member again,
23         *         set OPTION MRECALL.;
24         %Include Pgm(TestS);
27         %TestS
@@TESTS stored in file TestS.sas
28         %Include Pgm(ListMcat);
ListMcat: SASmStore=SiteMlib
ListMcat: List Macros in Catalog(s)
libname  objname  objdesc
SITEMLIB TESTS    TestS stored in file TestS
```

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

TestCompileNStoreX2.sas

```
Libname              StorProj '<dir-spec project>';
Libname              StorSite '<dir-spec site   >';
Libname              LibMacro (StorProj StorSite) ;
options SASmStore =  LibMacro  Mstored;
```

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.

CompStorSAS.sas

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

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

CompileAndStore.sas

```
 /*      name: CompileAndStore.sas
-------------: User Requirements:
      purpose: create catalog of compiled macros
  description: %include macro source code, compile and store
-------------: Program Specifications:
program group: maintenance of macro catalog
program  type: routine
    SAS  type: job
        input: macros in current directory:
               !SASsite\macroLibrary
      process: %Include each macro
       output: SASmacr.catalog
               log and list of compiled macros
        notes: required: macros must have option store
               option  : macros may  have option source: v9
       usage: batch submit
      author: Ronald.Fehd@cdc.hhs.gov
note: see CompStorSAS to load SAS-supplied core macros
change notes:
RJF2 05Jan31 for SUGI30 SASautos paper
/*..................................... */
filename               Macro    '.';
libname                SiteMacr '.';
options  SASmStore    = SiteMacr mStored
         McompileNote = all      MautoLocDisplay;

%Include Macro(Array);
%Include Macro(Nobs);
*Include Macro(...);

/*use utility from paper to show SASmaro.catalog **
filename Pgm 'L:\LaTeX\SUGI30';*'!SASsite\includes';
%Include Pgm(ListMcat)/nosource2;
%Include Pgm(ProcCatalogSASmStoreSASmacr)/nosource2;
%Include Pgm(ProcSQLSASmStoreSASmacr)/nosource2;
/************************************/
```

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:

ProcCatalogSASmStoreSASmacr.sas

```
/* name: ProcCatalogSASmStoreSASmacr.sas
purpose: list compiled and stored macros
***..................................... */
PROC Catalog
     catalog = %sysfunc(getoption(SASmStore)).SASmacr;
     contents;
     quit;
run;
```

ProcSQLSASmStoreSASmacr.sas

```
/* name: ProcSQLSASmStoreSASmacr.sas
purpose: list compiled and stored macros
***..................................... */
proc SQL;select *
         from   Dictionary.Catalogs
         where  MemName in ('SASMACR')
        ;quit;
run;
```

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

TestSinProgramA.sas

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

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.

TestCompileNStoreOK.log continued

```
107         %Include Pgm(TestSinProgramA);
110         %TestS
@@TESTS in file TestSinProgramA.sas
111         %Include Pgm(ListMcat);
ListMcat: SASmStore=SiteMlib
ListMcat: List Macros in Catalog(s)
libname  objname  objdesc
SITEMLIB TESTS    TestS stored in file TestS
WORK     TESTS    TestS in file TestSinProgramA
```

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 autocall feature and compilation and storage of macros.

autoexecExample.sas

```
* name: autoexecExample.sas;
Filename              Pgm      '.';
Libname               MacroLib '.';
options  SASautos  = (Pgm SASautos) MautoSource
         SASmStore =  MacroLib      Mstored;
```

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

AutoCallDemo.log

```
2          %Include Pgm(ListMcat);
ListMcat: SASmStore=MacroLib
ListMcat: List Macros in Catalog(s)
libname  objname  objdesc
MACROLIB TESTS    TestS stored in file TestS
80         %TestS
@@TESTS stored in file TestS.sas
81
82         %TestX
@@TESTX in file TestX.sas
83         %Include Pgm(ListMcat);
ListMcat: SASmStore=MacroLib
ListMcat: List Macros in Catalog(s)
libname  objname  objdesc
MACROLIB TESTS    TestS stored in file TestS
WORK     TESTX    TestX in file TestX
```

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:

autoexecCorp.sas

```
*name: a Corporate autoexec.sas;

FileName SASmacro %sysGet(sasAutos) ;

Options  sasAutos=("!ProjMacs"    /* project library, if any */
                   "L:\CorpMacro" /* corporate library       */
                   SASmacro       /* SAS macro libraries     */
                  )
         MautoSource;
```

AutoexecSite: The author uses this autoexec:

autoexecSite.sas

```
 /*      name: autoexecSite.sas
-------------: User Requirements:
      purpose: standardization of filename Pgm, SASautos
  description: concatenate site folders
                            with SAS-supplied folders
-------------: Program Specifications:
program group: invocation
program  type: routine
    SAS  type: include w/parms from configuration
         input: see project configuration file
       process: allocation of filerefs, librefs, options
        output: to log if invoked with -autoecho
         notes: calls project autoexec
         usage: -config u:\dls\SAS\sitev9?.cfg contains:
                -SET        SITEroot "u:\dls\SAS"
                -autoexec "!SITEroot\autoexecSite.sas"
        author: Ronald.Fehd@cdc.hhs.gov
note: project libref:Library allocated in project autoexec
note: %sysget(SiteRoot) could be !SiteRoot
change notes:
RJF2 02Jul31 original
RJF2 05Feb02 polish and trim for SUGI30 SASautos paper
*** ................................. */
%Put @@ -------- autoexecSite.sas --------;

*concatenate project directory with site includes;
Filename Pgm      ('.' "%sysget(SiteRoot)\includes");

*prepend project and site macros to SAS-supplied macros;
Filename SASautos ('.' "%sysget(SiteRoot)\macros"
                        %sysget(SASautos)
                   ;*correct: no close paren for filename;

Libname SiteMlib "!SITEroot\macroCat";

options dtReset              %*V9: date+time reset in title1;
        MautoSource          %*autocall := on;
        Mstored              %*compiled macros stored;
        SASMstore = SiteMlib %*in SASmacr.catalog;
        ;
%Put Pgm(AutoExec) begin; %Inc Pgm(AutoExec);
%Put Pgm(AutoExec) end;
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 ListMCat.sas is described in this paper:

Fehd, Ronald (2005), *Journeymen'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 greatful. 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 since1997
	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
```

PROGRAM ListMcat.sas

ListMcat.sas

```
 /*      name: ListMcat.sas
-------------: User Requirements:
      purpose: to show masked macros:
                  two same-named macros in different catalogs
  description: List Macros in Catalogs
               notes written to log
                  of LibName, MacroName, Description
-------------: Program Specifications:
program group: testing for masking: aka name collision
program  type: routine
    SAS  type: include w/o parms
        input: SAShelp.VCatalg
      process: read from SAShelp.vCatalg
               put  vars
       output: notes in log
        notes: see also macro PutMasked
                   in book: A SAS Companion
        usage: filename Pgm '.';
               %Include Pgm(ListMcat);
       author: Ronald.Fehd@cdc.hhs.gov
change notes:
RJF2 05Jan23 for SUGI30 SASautos paper
RJF2 05Jan30 compile&store MemName is always SASmacr
/*..................................... */
options nonotes;
%Let WidthObjName = 32;%*max width macro name::ObjName;
%Let WidthObjName =  7;%*for demo: length(objname) = 7;
DATA _Null_;
file LOG;%*ruler for column value assignments;
*   .    1    .    2    .    3    .    4    .    5
.  .5.  .0.  .5.  .0.  .5.  .0.  .5.  .0.  .5.  .01234
ListMcat: List Macros in Catalog(s)
libname8 objname8901234567890123456789012  objdesc
LIBRARY  TESTY                             TestY
;*end ruler;
retain SASmStore "%sysfunc(getoption(SASmstore))"
       C1 1
       C2 10
       C3 44;
       C3 = sum(C2,&WidthObjName.,2);
if               SASmstore ne ' ' then
put  'ListMcat: ' SASmStore=;
put  'ListMcat: List Macros in Catalog(s)'
  /  @C1 'libname'
     @C2 'objname'
     @C3 'objdesc';
do   until(EndoFile);
set  SAShelp.VCATALG(where = (MemName eq 'SASMACR'))
                     end   = EndoFile;
Put  @C1 LibName
     @C2 ObjName
     @C3 ObjDesc;    end;         stop;
run;%SymDel WidthObjName;options notes;run;
```

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

Return to First Page Close Document