Paper BB-10-2015

An Autoexec Companion, Allocating Location Names during Startup

Ronald J. Fehd, Stakana Analytics

Abstract

Description : Like other computer languages SAS® software provides a method to automatically execute statements during startup of a program or session. This paper examines the names of locations chosen in the filename and libname statements and the placement of those names in options that enable all programs in a project to have standardized access to format and macro catalogs, data sets of function definitions and folders containing reusable programs and macros. It also shows the use of the global symbol table to provide variables for document design.

Purpose : The purpose of this paper is to examine the default values of options, suggest naming conventions where missing, and provide both an example autoexec and a program to test it.

Audience : programmers, all levels

Keywords : document design (`title, footnote`),
environment and macro variables,
file searchs: library (*librefs*), programs (*filerefs*),
format search (`fmtsearch`),
functions (`cexist, exist, ifc, getoption, %sysfunc`),
macro autocall: file search (`mautosource, sasautos`),
macros, catalog of compiled and stored (`mstored, sasmstore`),
procedure `fcmp` data set of functions (`cmplib`)

In this paper

Introduction

Overview

SAS software provides a facility to automatically execute statements during startup of a program. The option name is autoexec. Its default value is `autoexec.sas`, in the Windows operating system. If a file with this name is present in the startup folder then it is included before the program.

The autoexec file provides all programs in the project folder the same list of asset names and locations. Names of locations are allocated with the `filename` and `libname` statements. These location names, referred to as *filerefs* and *librefs* respectively, are arbitrary. Whichever names are chosen, the next step is to insert them into the corresponding option which enables SAS software to search for a particular object within the asset.

Options and Location Names

asset	option	value	default	object	in	see also
functions	`cmplib`	*libref.data*	n/a	definitions	data set	
formats	`fmtsearch`	*librefs*	work library	formats	catalog	`fmterr`
macros						
autocall	`sasautos`	*filerefs*	`sasautos`	files	folder	`mautosource`
compiled	`sasmstore`	*libref*	n/a	definitions	catalog	`mstored`

The introduction has these topics.

- style guide
- naming conventions
- definitions
- folder structure
- syntax for lists

style guide

This paper uses *italics* and `text` fonts to identify specific uses of words.

italics : SAS documentation words, e.g. *fileref*, *libref*

`text` : SAS keyword, e.g. `filename`, `libname`

naming conventions

This is the list of naming conventions used in this papers

site : folders available to all projects are referred to with a prefix of 'site_'

project : the 'logical name' for the folder containing *.sas programs is 'project'

I/O : other folders have a prefix according to function:
input: 'in_', output: 'out_',

definitions

This is a list of the terms used here.

addressing : a method of referring to another folder
- absolute : provide a full (direct) path to the folder using either a drive letter or universal-naming convention (UNC), e.g., `'c:\temp'` — also known as 'hard-coded'
- relative : provide an indirect reference to a sibling or parent folder e.g., `'..\csv'` — also known as 'soft-coded'

see paragraph 'folder structure' below

directory : see folder; in SAS documentation, see 'directory-specification'

fileref : is a 'logical name' for one or more folders; first argument of the `filename` statement; the value of a *fileref* is also called a 'directory-specification'

folder : a folder contains files and may contain sub-folders (or child-folders); as used in this paper folders contain text files; also called a directory

library : is one or more folders; in SAS documentation 'library' generally refers to folder(s) containing files maintained by SAS software, i.e., catalogs and data sets

libref : is a 'logical name' for a library, first argument of the `libname` statement, the value of a *libref* is also called a 'directory-specification'

'logical name' : is a phrase used in SAS documentation; here it means a variable name in the global symbol table

---

folder structure

The assignments discussed in this paper are based on these assumptions; your corporate culture may differ.
The purpose of this structure is to support copying of programs from a working project to a new project and minimize changes necessary to get the programs working.

```
...\project-A\
                csv             comma separated values
                dat             data files
                out-pdf         vendor-specific files
                out-rtf         vendor-specific files
                sas             *.sas programs
                sas7b           data sets, catalogs
...\project-B\  like project-A
...\SAS-site\
                includes        *.sas programs
                macros          *.sas programs
                sas7b           data sets, catalogs
```

---

syntax for lists

A list contains one or more items. In the `filename`, `libname` and `options` statements a value may be a single item or a list of items which must be enclosed in parentheses, e.g.,
item: `filename project '.';` — one folder
list: `options sasautos = (project site_mac sasautos);` — three *filerefs*

Programs for Discovery during Research

Overview

These programs were used in research for this paper.

- options define value
- options groups
- sql dictionary options
- getoption: fetching value
- finding word in options

options define value

The `define` and `value` options are used to display information about an option in the `options` procedure.

```
PROC options define value option = autoexec;
```

options groups

This program shows the groups of the options discussed here.

```
PROC options group = envfiles;*autoexec fmtsearch sasautos;
PROC options group = macro   ;*mautosource mstored sasmstore;
PROC options group = sasfiles;*cmplib;
```

sql dictionary options

The `sql` procedure offers another avenue of discovery.

```
PROC sql; describe table dictionary.options;
          quit;
PROC sql; select distinct group
          from dictionary.options;
          quit;
```

getoption: fetching value

The `getoption` function can be called in open code by the `%sysfunc` function. There are three types of options: boolean, char and num. Character and numeric option values can be fetched with the `keyword` option. This returns the value in the form: `opttion-name=value`. This value can be used in a macro variable assignment statement.

```
**** option type character;
%let %sysfunc(getoption(autoexec,keyword));
%put echo &=autoexec;
**** option type boolean in (T=<opt-name>,F=no<opt-name>);
%put mautosource=%sysfunc(getoption(mautosource));
options nomautosource;
%put mautosource=%sysfunc(getoption(mautosource));
```

log:

```
echo AUTOEXEC=C:\...\autoexec-companion\sas\autoexec.sas
5    %put mautosource=%sysfunc(getoption(mautosource));
mautosource=MAUTOSOURCE
6    options nomautosource;
7    %put mautosource=%sysfunc(getoption(mautosource));
mautosource=NOMAUTOSOURCE
```

---

finding word in options

The `sql` procedure can be used to search for a word in either the option name or the option description.

```
%let word = macro;
PROC sql; select group, optname, optdesc
          from   dictionary.options
          where  index(upcase(optname),"%upcase(&word)")
             or  index(upcase(optdesc),"%upcase(&word)");
          quit;
```

Allocating Variables in the Global Symbol Table during Startup

Overview

This section describes the allocations of variable names and values in the global symbol table that are used in programs.

As used in this paper, the term 'global symbol table' has these entities.

- environment variables
- filerefs location names
- librefs location names
- macro definitions
- macro variables
- options
- running text: titles and footnotes

Global Variables for Output

Overview

Managing output has two aspects, marking up the document and where to write the document, i.e., the destination folder.

- document design
- output destination

document design

Output documents may have running text, which consists of either or both of headers or footers. The global symbol table contains a maximum of ten entries for `title` or `footnote` assignments.

naming conventions → The naming conventions used here are `title1` contains the project name and `title2` contain the name of the program; titles3–10 are assigned and cleared by programs.

```
title1 'An autoexec.sas Companion';
       *** sysin contains path\program-name.sas;
title2 "%sysfunc(getoption(sysin))";
```

output destination

All output may be directed to a single folder with the use of an environment variable. A bang or exclamation point (!) is used to refer to an environment variable.

```
******** allocate environment variables;
         *** relative: sibling folder;
options  set = out_pdf '..\out-pdf';
```

usage:

```
ods pdf file = '!out_pdf\report.pdf';
```

naming conventions → Refer to the paragraph 'folder structure' above.

commentary Other authors use macro variables for this task.

Filerefs for Input, Programs And Autocall Macros

Overview

One important use of the global symbol table is for names of locations of reusable programs and input text. This section provides suggestions for naming of the following items.

- filerefs for input
- filerefs for programs
- autocall macros

---

filerefs for input

Use 'in_' as the prefix of names for folders containing text files of input. These example shows the folders are sibling folders and are described with relative addressing.

```
filename in_csv '..\csv';

filename in_dat '..\dat';
```

naming conventions → Refer to the paragraph 'folder structure' above.

---

filerefs for programs

Project programs may reuse other programs in the project folder as well as utility programs in the site folder.

naming conventions → Use 'site_' as the prefix of location names in the site folder.

```
filename project  '.';                 * here: startup folder;

         *** allocate environment variables (e-var);
                                       * absolute: hard-coded;
options  set = site_root 'C:\SAS-projects\SAS-site';
         *** use bang (!) as prefix to reference an e-var;
filename site_inc "!site_root\includes\";  *   double quotes;
filename site_mac '!site_root\macros\'  ;  *or single quotes;
```

Notes:

- This example shows the environment variable 'site_root' is hard-coded with a path to a folder that is not a sibling of the project folder.
- Either single or double quotes may be used.

---

autocall macros

There are a pair of options for *autocall* macros;
`mautosource` is boolean, `sasautos` is character and is a list of *filerefs*.
This `options` statment sets up the *autocall* macro search facility for three *filerefs*: `project`, `site_mac`, and the SAS-supplied *fileref* `sasautos`.

```
** pair  macro autocall: boolean, search-path filerefs;
options  mautosource
         sasautos = (project site_mac sasautos);
```

Note: The list of filerefs must be enclosed in parentheses.

Librefs for Format and Macro Catalogs, Data Sets and Functions

Overview

This section describes naming convention choices and statements to allocate location names for SAS-managed files such as catalogs of formats, compiled macros, and data sets containing functions.

- choose libref library
- catalogs for formats
- catalogs for macros
- data sets for fcmp functions

---

choose libref library

There are two reasons to use the *libref* `library`.

1. SAS uses the term throughout its documentation
2. the default search path for format catalogs is `library`

---

catalogs for formats

The format procedure compiles `value` statements and other definitions and saves into the the *libref* named in the `library` option of the procedure statement.

syntax : `libname` *libref* `'folder';`

usage : `libname library '..\sas7b'; * sibling folder;`

Note: The statement `proc format library = library;` creates a catalog in the *libref* `library`.

naming conventions → The catalog name is `library.formats`.
Libref `library` is the second *libref* in the default format-search path.

Examine the format search path with this statement.

```
%put %sysfunc(getoption(fmtsearch,keyword));
fmtsearch=(WORK LIBRARY)
```

beware naming collisions

Do not change the format search path to `(library work)` because:

- it is frustrating for users to create formats, which are stored in the *libref* `work`, that are superseded by formats in the *libref* `library`
- it is extremely difficult to debug this problem

Note: The format catalogs in *librefs* `work` and `library` are always searched even if they are not in the `fmtsearch` list.

---

catalogs for compiled macros

Macros may be compiled and stored in a catalog.
Use the `macro` statement option `store` to save in a catalog.

example:

```
%macro demo(...)/store;
```

There are a pair of options for compiled and stored macros;
`mstored` is boolean, `sasmstore` is character and is one *libref*.

```
libname  site_lib '!site_root\sas7b';
** pair  macros compiled and stored: boolean, libref;
options  mstored  sasmstore = site_lib;* or library;
```

Note: The name of the catalog where compiled and stored macro definitions are saved is *libref.sasmacr*. The search sequence is *libref* `work` where autocall macros are saved, then *libref* of option `sasmstore`; this search sequence cannot be changed. Compare to the option `fmtsearch`.

naming conventions → Be aware of naming collisions when using compiled and stored macros, i.e., macros with the same name in both the `work` and `sasmstore` catalogs.

---

data set for fcmp functions

There are a pair of statements for functions created by the `fcmp` procedure.
`libname` allocates a *libref*; option `cmplib` is character;
it is a two-level data set name, e.g., *libref.data*.

naming conventions → Both the *libref* and data set name are arbitrary.

```
*creation:  proc fcmp outlib=libref.data_set_name.package;
******* functions compiled by proc fcmp;
options cmplib = site_lib.functions;  *or library.functions;
```

Other Options

Overview

There are several hundred options many of which may be set in the autoexec. See the paragraph Suggested Reading below for other authors' recommendations.

```
options  fullstimer       %* max info in timer block;
         msglevel = i     %* more messages in log   ;
         nofmterr;         * no err: formats missing;
```

---

append, insert

Many of the above options can be modified with either or both of the options `append` or `insert`.

```
*syntax: options insert|append=(opt-name=libref|'..\path');
*example;
options insert=(fmtsearch=lib_a);*libref;
*discovery;
proc options option=fmtsearch value;
```

Note: In SAS documentation see the pages *APPEND= System Option*, *INSERT= System Option*, and *Using SAS System Options*, paragraphs: *Option Value Information for an Option Modified by the INSERT and APPEND Options* and *Changing an Option Value by Using the INSERT and APPEND System Options*.

Programs

Overview

This section contains the listings of an `autoexec.sas` and test programs.

autoexec.sas

```
title1   'An autoexec.sas Companion';
         %let %sysfunc(getoption(sysin,keyword));
         %put echo &=sysin;
title2   "&sysin";               *** path\program-name.sas;
title2   "%scan(&sysin,-2,/\.)"; *** program-name only;

         *** allocate environment variables;
               *** relative: sibling folder;
options  set = out_pdf   "..\out-pdf";    *   double quotes;
options  set = out_rtf   '..\out-rtf';    *or single quotes;
               *** absolute: hard-coded   compare to site_lib;
options  set = site_root 'C:\SAS-projects\SAS-site';

         *** various folders with input files;
filename in_csv  '..\csv';
filename in_dat  '..\dat';
filename project '.';* here: startup folder: *.sas programs;
         *** use bang (!) as prefix to reference an e-var;
filename site_inc "!site_root\includes";  *   double quotes;
filename site_mac '!site_root\macros'  ;  *or single quotes;

libname  library  '..\sas7b';               *sibling folder;
*libname  site_lib '!site_root\sas7b';
libname  site_lib '..\..\SAS-site\sas7b';*compare to site_root;

         *** functions compiled by proc fcmp in data set;
options  cmplib = library.functions;*or site_lib.functions;

         *** format catalogs, see also option fmterr;
options  fmtsearch = (work library site_lib.formats);

** pair  macro autocall: boolean, search-path filerefs;
options  mautosource
         sasautos = (project site_mac sasautos);

** pair  macros compiled and stored: boolean, one libref;
options  mstored  sasmstore = library;      *or site_lib;

options  nofmterr         %* no err msg: fmt missing;
         fullstimer       %* max info in timer block;
         msglevel = i     %* more messages in log   ;
```

Autoexec-test.*

Overview

The autoexec-test program uses the following functions.

`cexist` : exist catalog

`exist` : exist libref.data-set

`getoption` : returns value of an option

`ifc` : evaluate condition, return character string

`%nrstr` : no rescan string, delays evaluation

`%sysfunc` : system function, allows access to certain data step functions; used here with `ifc` to conditionally execute statements

`%sysfunc(ifc` :
```
%sysfunc(ifc(condition
             ,%nrstr(%put true ;)
             ,%nrstr(%put false;) ))
```
evaluate a condition in open code, return either true statement(s) or false statement(s);
the `%nrstr` function guarantees only one statement is returned

`%let catalog=` : macro variable assignment statement

`exist_catalog` :
```
%let exist_catalog = %nrstr(echo
%sysfunc(ifc(%sysfunc(cexist(&catalog))
      ,,%nrstr(not ),))exist &catalog);
```
this macro variable contains a function call to verify the existence of the catalog named in the macro variable `catalog`

`%unquote` :
```
%let catalog = work.formats;
%put %unquote(&exist_catalog);
```
resolves the macro variable references and function calls in the macro variable `exist_catalog`

sas.bat

This batch file calls SAS software.

```
call "C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" %*
```

The DOS command 'call' allows multiple instances in another batch file. The string percent+asterisk '%*' passes all command-line parameters on to `sas.exe`.
See usage in `autoexec-test.bat` below.

sasv9.cfg

```
/* name: <project-name>\sas\sasv9.cfg */
-sasinitialfolder '.' /* set 'file-open' to startup folder */
-nosplash             /* no display SAS banner at startup  */
-noovp                /* no overprint errors nor warnings  */
```

Without this project configuration file the *File-Open* dialogue defaults to
`"C:\Documents and Settings\<user-name>\My Documents\My SAS Files\9.4"`

autoexec-test.bat

```
sas autoexec-test -echoauto
```

autoexec-test.sas

```
**** name: autoexec-test.sas;
**** many options are set in configuration files;
%put echo sascfg=%sysget(sascfg);
%put echo %sysfunc(getoption(config,keyword));

*command-line option sasinitialfolder may be set by icons;
*or project config, compare to folder of fileref project;
%put echo %sysfunc(getoption(sasinitialfolder,keyword));

*review all filerefs;
filename _all_ list;

**** review cmplib=libref.data-set-name of functions;
%let %sysfunc(getoption(cmplib,keyword));
%sysfunc(ifc(%sysfunc(exist(&cmplib))
           ,%nrstr(%put echo exist(&cmplib);)
        ,%nrstr(%put echo not exist(&cmplib);),))

%let exist_catalog = %nrstr(echo
%sysfunc(ifc(%sysfunc(cexist(&catalog))
      ,,%nrstr(not ),))exist &catalog);

**** formats: search list of librefs;
%let %sysfunc(getoption(fmtsearch,keyword));*remove parens;
%let fmtsearch=%substr(&fmtsearch,2,%length(&fmtsearch)-2);
%put echo &=fmtsearch;
**** check default = work library;
%let catalog = work.formats;
%put %unquote(&exist_catalog);
%let catalog = library.formats;
%put %unquote(&exist_catalog);
**** remove default librefs WORK and LIBRARY;
%let fmtsearch=%sysfunc(tranwrd(&fmtsearch,WORK,%str( )));
%let fmtsearch=%sysfunc(tranwrd(&fmtsearch,LIBRARY,%str( )));
**** there may be one, or more;
%let fmt_lib = %scan(&fmtsearch,1,%str( ));
**** if fmt-lib has dot then is two-level name;
%let catalog = %sysfunc(ifc(%index(&fmt_lib,.)
     ,%nrstr(&fmt_lib),%nrstr(&fmt_lib..formats) ));
%put %unquote(&exist_catalog);

**** macro autocall;
%put echo mautosource=%sysfunc(getoption(mautosource));
%put echo %sysfunc(getoption(sasautos,keyword));
*allocate fileref sasautos to check if modified in config;
%include sasautos(af)/nosource2;
%put echo sasroot=%sysget(sasroot);
filename sasautos list;
**** if  sasautos contains paths other than sasroot
     then review configuration file(s) listed above;

**** macros compiled and stored;
%let mstored  =%sysfunc(getoption(mstored));
%let sasmstore=%sysfunc(ifc(&mstored eq MSTORED
     ,%nrstr(%sysfunc(getoption(sasmstore))),%nrstr(work)));
%put echo &=mstored &=sasmstore;
%let catalog = &sasmstore..sasmacr;
%put %unquote(&exist_catalog);
```

autoexec-test-log.txt

```
NOTE: AUTOEXEC processing beginning; file is
      C:/.../autoexec-companion/sas/autoexec.sas.
19    libname library '..\sas7b';
NOTE: Libref  LIBRARY was successfully assigned as follows:
      Physical Name: C:\...\autoexec-companion\sas7b
NOTE: AUTOEXEC processing completed.
1     **** name: autoexec-test.sas;
2     **** many options are set in configuration files;
3     %put echo sascfg=%sysget(sascfg);
echo: sascfg=C:\...\SASHome\SASFoundation\9.4\nls\en
4     %put echo %sysfunc(getoption(config,keyword));
echo: CONFIG=("C:\...\SASHome\SASFoundation\9.4\sasv9.cfg"
              "C:\...\SASHome\SASFoundation\9.4\nls\en\sasv9.cfg"
              "C:\...\autoexec-companion\sas\sasv9.cfg" )
8     %put echo %sysfunc(getoption(sasinitialfolder,keyword));
echo: SASINITIALFOLDER=.
11    filename _all_ list;
...
NOTE: Fileref= PROJECT
      Physical Name= C:\...\autoexec-companion\sas
NOTE: Fileref= SITE_INC
      Physical Name= C:\SAS-projects\SAS-site\includes
NOTE: Fileref= SITE_MAC
      Physical Name= C:\SAS-projects\SAS-site\macros
13    **** review cmplib=libref.data-set-name of functions;
14    %let %sysfunc(getoption(cmplib,keyword));
15    %sysfunc(ifc(%sysfunc(exist(&cmplib))
16        ,%nrstr(%put echo exist(&cmplib);)
17    ,%nrstr(%put echo not exist(&cmplib);),))
echo: not exist(library.functions)
19    %let exist_catalog = %nrstr(echo
20    %sysfunc(ifc(%sysfunc(cexist(&catalog))
21          ,,%nrstr(not ),))exist &catalog);
23    **** formats: search list of librefs;
24    %let %sysfunc(getoption(fmtsearch,keyword));*remove parens;
25    %let fmtsearch=%substr(&fmtsearch,2,%length(&fmtsearch)-2);
26    %put echo &=fmtsearch;
echo: FMTSEARCH=WORK LIBRARY SITE_LIB.FORMATS
27    **** check default = work library;
28    %let catalog = work.formats;
29    %put %unquote(&exist_catalog);
echo: not exist work.formats
30    %let catalog = library.formats;
31    %put %unquote(&exist_catalog);
echo: not exist library.formats
32    **** remove default librefs WORK and LIBRARY;
33    %let fmtsearch=%sysfunc(tranwrd(&fmtsearch,WORK,%str( )));
34    %let fmtsearch=%sysfunc(tranwrd(&fmtsearch,LIBRARY,%str( )));
35    **** there may be one, or more;
36    %let fmt_lib = %scan(&fmtsearch,1,%str( ));
37    **** if fmt-lib has dot then is two-level name;
38    %let catalog = %sysfunc(ifc(%index(&fmt_lib,.)
39        ,%nrstr(&fmt_lib),%nrstr(&fmt_lib..formats) ));
40    %put %unquote(&exist_catalog);
echo: not exist SITE_LIB.FORMATS
42    **** macro autocall;
43    %put echo mautosource=%sysfunc(getoption(mautosource));
echo: mautosource=MAUTOSOURCE
44    %put echo %sysfunc(getoption(sasautos,keyword));
echo: SASAUTOS=(project site_mac sasautos)
...
72    %put echo sasroot=%sysget(sasroot);
echo: sasroot=C:\...\SASHome\SASFoundation\9.4
73    filename sasautos list;
NOTE: Fileref= SASAUTOS
      Physical Name= C:\...\SASHome\SASFoundation\9.4\core\sasmacro
...
      Physical Name= C:\...\SASHome\SASFoundation\9.4\stat\sasmacro
77    **** macros compiled and stored;
78    %let mstored  =%sysfunc(getoption(mstored));
79    %let sasmstore=%sysfunc(ifc(&mstored eq MSTORED
80    ,%nrstr(%sysfunc(getoption(sasmstore))),%nrstr(work) ));
81    %put echo &=mstored &=sasmstore;
echo: MSTORED=MSTORED SASMSTORE=library
82    %let catalog = &sasmstore..sasmacr;
83    %put %unquote(&exist_catalog);
echo: not exist library.sasmacr
```

Summary

Conclusion

An autoexec file provides the ability to centralize statements which manage entries into and relations within the global symbol table. This centralization encourages standardization and reduces the complexity, and therefore the testing burden, of all programs in the project.

Suggested Reading

predecessor : Fehd [6] SASautos Companion, Fehd [5]

autoexec : Cogswell [3] describes a robust set of Windows batch programs that uses `systask` with error checking and notification, shows directory structure, how to restart, provides method of debugging (testing) individual programs.
Fang and Gorrell [4] discuss use of macro variables
and the `%include` statement;
Carpenter and Smith [2] discuss naming conventions of directories, files and libraries; use macro variables for paths, production/testing and titles; provide paths in option `sasautos`; shows two *librefs* for compiled-and-stored macro option `sasmstore`
Taubman [11] shows `filename` statement with `pipe` option in an autoexec, uses environment variables

batch : Walsh [13] shows differences between program development in session vs. batch and why an autoexec is necessary for batch

formats : Gudavalli [7]

functions : user-written functions, `proc fcmp`, Carpenter [1], Secosky [10]

libname : William E. Benjamin Jr [14]

options : Heaton [8], Poll [9], Thornton and Barbalau [12]

Contact Information: **Ronald J. Fehd** `mailto:Ron.Fehd.macro.maven@gmail.com`
`http://www.sascommunity.org/wiki/Ronald_J._Fehd`

About the author:

education:	B.S. Computer Science, U/Hawaii, 1986
	SAS User Group conference attendee since 1989
experience:	programmer: 30+ years
	author: 40+ SUG papers
	sas.community.org wiki: 400+ pages
SAS-L:	author: 7,000+ messages to SAS-L since 1997

Programs: **http://www.sascommunity.org/wiki/Autoexec_Companion**

Trademarks

Bibliography

[1] Arthur L. Carpenter. Using proc fcmp to the fullest: Getting started and doing more. In *SAS Global Forum*, 2013. URL support.sas.com/resources/papers/proceedings13/139-2013.pdf. 10 pp.; user-written functions, routines and subroutines with proc fcmp, arguments, returning multiple character or numeric values, using sysfunc to call.

[2] Arthur L. Carpenter and Richard O. Smith. Library and file management: Building a dynamic application. In *SUGI-27*, 2002. URL http://www2.sas.com/proceedings/sugi27/p021-27.pdf. 8 pp., ApDev; autoexec, directory naming conventions, macro libraries, filerefs and librefs;.

[3] Denis Cogswell. More than batch – a production SAS(R) framework. In *SUGI-30*, 2005. URL http://www2.sas.com/proceedings/sugi30/021-30.pdf. 15 pp.; ApDev, autoexec in production environment.

[4] Zhengyi Fang and Paul Gorrell. Two steps to libname-free coding: Use of macro parameters and autoexec.sas. In *NESUG*, 2010. URL http://www.lexjansen.com/nesug/nesug10/ad/ad03.pdf. 5 pp.; using global macro variables to replace libref.data-set names.

[5] Editor R.J. Fehd. Sasautos companion. In *sas community.org Wikipedia*, 2007. URL http://www.sascommunity.org/wiki/SASautos_Companion_Reusing_Macros.

[6] Ronald Fehd. A SASautos Companion: Reusing macros. In *Proceedings of the 30th Annual SAS® Users Group International Conference*, 2005. URL http://www2.sas.com/proceedings/sugi30/267-30.pdf. Tutorials, 12 pp.; topics: autocall, autoexec, configuration file (sasv9.cfg), compiled and stored macros, masking, options (mautosource, mstored, sasmstore), program reuse, sasautos (environment variable, filename, option); info: autoexec examples, utility program ListMcat: show same-named macros in different catalogs.

[7] Ram Gudavalli. Search across format libraries in FMTSEARCH option. In *NESUG*, 2011. URL http://www.lexjansen.com/nesug/nesug11/pf/pf03.pdf. 3 pp.; list or verify formats in various catalogs.

[8] Edward Heaton. SAS(R) system options are your friends. In *SESUG*, 2004. URL http://analytics.ncsu.edu/sesug/2004/TU07-Heaton.pdf. 15 pp.; nofmterr, sasinitialfolder.

[9] Denise Poll. Options — versatile players in the game of SAS. In *SAS Global Forum*, 2011. URL http://support.sas.com/resources/papers/proceedings11/240-2011.pdf. 8 pp.; Programming, Beyond Basics; system tools available to view and manage options, procedures optload and optsave, options modifiable by insert and append.

[10] Jason Secosky. Executing a proc from a data step. In *SAS Global Forum*, 2012. URL http://support.sas.com/resources/papers/proceedings12/227-2012.pdf. 12 pp.; user-written functions with proc fcmp, functions run_macro, run_sasfile, dosub, dosubl; proc http.

[11] Stephen B. Taubman. Procedures for a new SAS(R) release. In *SUGI-25*, 2000. URL http://www2.sas.com/proceedings/sugi25/25/po/25p206.pdf. 3 pp.; autoexec for site, filename with pipe option.

[12] Patrick Thornton and Iuliana Barbalau. Working the system: Our best SAS(R) options. In *SAS Global Forum*, 2012. URL http://support.sas.com/resources/papers/proceedings12/047-2012.pdf. 10 pp.; tools to examine options.

[13] Irina Walsh. Pros and cons of interactive SAS(R) mode vs. batch mode. In *WUSS*, 2010. URL http://www.wuss.org/proceedings10/coders/2937_4_0_COD_Walsh.pdf. 9 pp.; uses macro variables in autoexec.sas, directory structure, sysfunc+ifc, proc printto for log and output in session, ods rtf.

[14] William E. Benjamin Jr. The little engine that could: Using libname engine options to enhance data transfers between SAS(R) and Microsoft Excel files. In *SESUG*, 2011. URL http://analytics.ncsu.edu/sesug/2011/CC13.Benjamin.pdf. 10 pp.;.

Information is *the* difference
that makes *a* difference

Gregory Bateson, 1904–1980
Steps to an Ecology of Mind, 1972