

Paper CC-187

Macro Code to Test Existence of Various Objects

Ronald J. Fehd, Stakana Analytics

Abstract	Description: SAS[®] software provides functions to check the existence of the objects which it manages — catalogs and data sets — as well as folders referred to by the <code>filename</code> and <code>libname</code> statements. Purpose: The purpose of this paper is to provide a set of macro statements for assertions of existence and to highlight the exceptions where these functions return non-boolean choices. Audience: programmers writing macros Keywords: macro functions <code>%sysfunc</code> and <code>ifc</code>; data: <code>exist</code>, catalog: <code>cexist</code>, filename: <code>fileexist</code>, <code>fileref</code> of file: <code>fexist</code>, <code>fileref</code> of folder: <code>fileref</code>, <code>libref</code>: <code>libref</code>
-----------------	---

In this paper	Open Code	1
	Macros	4
	Summary	8
	References	9

Introduction	Every programmer has a tool box, a collection of often-used statements that they use when writing new programs. This is a collection of the author's tools.
---------------------	---

Open Code

Overview	This is the overview, which consists of a list of topics in this section. • <code>%sysfunc</code> • verify functions • <code>ifc</code> • catalog • data • filename • <code>fileref</code> of file • <code>fileref</code> of folder • <code>libref</code>
-----------------	--

%sysfunc Macro function %sysfunc has one required argument, a function call; the function may have one or more arguments.

```
1 %sysfunc(<function(...)>)
```

Notes: This is not a macro statement, it is a function call and does not need a semicolon for closure.

verify functions

cexist :	catalog or catalog entry	%sysfunc(cexist(catname))
exist :	library member	%sysfunc(exist(libref.memname))
fileexist :	filename.ext	%sysfunc(fileexist(filename.ext))
fexist :	fileref of file	%sysfunc(fexist(fileref-file))
fileref :	fileref of folder	%sysfunc(fileref(fileref-folder))
libref :	libref	%sysfunc(libref(libref))

ifc The ifc function has three arguments, logical expression, value returned when true, value for false. The %nrstr function is necessary and is used to delay macro resolution of the true and false values until the ifc function completes.

```
1 %sysfunc(ifc(<logical expression>
2             ,%nrstr(<true token(s)>)
3             ,%nrstr(<false token(s)>)
4             ) )
```

True tokens can be complete statements.

```
1 %sysfunc(ifc(<logical expression>
2             ,%nrstr(%put true statement(s);)
3             ,%nrstr(%put false statement(s);) ))
```

True tokens can be tokens within a statement.

```
1 %put info:
2 %sysfunc(ifc(<logical expression>
3             ,%nrstr(1=true)
4             ,%nrstr(0=false) )) message;
```

catalog This example show ifc returning tokens within a %put statement.

```
1 %let catalog = sashelp.stat;
2 %put
3 %sysfunc(ifc(%sysfunc(cexist(&catalog))
4             ,%nrstr(info: exist)
5             ,%nrstr(fail: not exist)
6             )) catalog(&catalog);
```

data This example show ifc returning a complete %put statement, and, in the case of failure, an endsas statement.

! → This is the recommended usage.

```

1 %let data = sashelp.class;
2 %sysfunc(ifc(%sysfunc(exist(&data))
3             ,%nrstr(%put info: exist data(&data);)
4             ,%nrstr(%put fail: not exist data(&data);
5             endsas;) ))

```

filename This example shows use a trivial macro function.

```

1 %macro exist_filename(filename);
2 %sysfunc(fileexist(&filename))
3 %mend;
4 %let filename = autoexec.sas;
5 %put
6 %sysfunc(ifc(%exist_filename(&filename)
7             ,%nrstr(info: exist)
8             ,%nrstr(fail: not exist))) file-name(&filename);

```

fileref of file This is the first of two examples which test existence of a *fileref*; this *fileref* is for a file.

```

1 filename          autoexec 'autoexec.sas';
2 %let fileref_file = autoexec;
3 %put %sysfunc(ifc(%sysfunc(fexist(&fileref_file))
4                 ,%nrstr(info: exist)
5                 ,%nrstr(fail: not exist))) file-ref(&fileref_file);

```

fileref of folder This is the second of two examples which test existence of a *fileref*; this *fileref* is for a folder.

```

1 %let fileref_folder = project;
2 %put
3 %sysfunc(ifc(not %sysfunc(fileref(&fileref_folder))
4             ,%nrstr(info: exist)
5             ,%nrstr(fail: not exist))) file-ref(&fileref_folder);

```

Notes: The logical expression requires negation since the *fileref* function returns zero for success.

libref The default value of the *fmtsearch* option is (work library).

```

1 %put info: fmtsearch: %sysfunc(getoption(fmtsearch));
2 %let libref = library;
3 %put
4 %sysfunc(ifc(not %sysfunc(libref(&libref))
5             ,%nrstr(info: exist)
6             ,%nrstr(fail: not exist))) lib-ref(&libref);

```

Notes: The logical expression requires negation since the *libref* function returns zero for success.

Macros

Overview

This section contains these topics.

- macro exist.sas
- program exist-test.sas
- listing exist-test.log
- macro callmacro excerpt

macro exist.sas

Macro exist centralizes all the function calls.

```

1  /*    name: exist.sas
2  description: returns boolean value in (0,1)
3              of existence of various objects
4  purpose: centralize existence tests
5  note: fileref and libref return -1,0,+1
6        zero is true, non-zero is false */
7  %macro exist
8      (data          = /*    exist */
9      ,catalog       = /*    cexist */
10     ,filename      = /*fileexist */
11     ,file_fileref  = /*    fexist */
12     ,fileref       = /*    fileref */
13     ,libref        = /*    libref */
14     ,testing = 0
15     )/des = 'demo: exist(any object)?' ;
16  %let testing = %eval(0 ne &testing
17     or %sysfunc(getoption(mprint)) eq MPRINT);
18
19  %if          %length(&catalog)      %then
20     %sysfunc(  cexist(&catalog));
21  %else %if    %length(&data)         %then
22     %sysfunc(  exist(&data));
23  %else %if    %length(&filename)     %then
24     %sysfunc(fileexist(&filename));
25  %else %if    %length(&file_fileref) %then
26     %sysfunc(  fexist(&file_fileref));
27  %else %if    %length(&fileref)      %then %do;
28     /*zero: fileref (+ external file) exist;
29     %eval(not %sysfunc(fileref(&fileref)));
30     %end;
31  %else %if %length(&libref)          %then %do;
32     /*if libref has been assigned
33     libref function returns 0
34     else returns nonzero value: -1, +1;
35     %eval(not %sysfunc(libref(&libref)));
36     %end;
37  %mend;
```

program exist-test.sas

Macro exist is tested with this program.

```

1  /* name: exist-test.sas
2  notes:
3  work.sasmacro catalog exists and contains macro exist
```

```

4   autoexec.sas  allocates filenames project and site_inc
5   turns on autocall:
6   options mautosource sasautos = (project sasautos);
7   **/
8   %put echo: %sysfunc(getoption(sasautos));
9   *include project(exist)/source2;
10  options mprint;
11  %let data = sashelp.class;
12  %put exist(&data)=%exist(data=&data);
13
14  %let data = work.class;
15  %put exist(&data)=%exist(data=&data);
16
17  %let filename = autoexec;
18  %put file-exist(&filename)=%exist(filename=&filename);
19
20  %let filename = autoexec.sas;
21  %put file-exist(&filename)=%exist(filename=&filename);
22
23  filename autoexec 'autoexec.sas';
24  %let file_fileref = autoexec;
25  filename &file_fileref list;
26  %put fileref-exist(&file_fileref)=%exist(file_fileref=&file_fileref);
27
28  %let fileref = work;
29  filename &fileref list;
30  %put f-exist(&fileref)=%exist(fileref=&fileref);
31
32  %let fileref = project;
33  filename &fileref list;
34  %put f-exist(&fileref)=%exist(fileref=&fileref);
35
36  %let fileref = site_inc;
37  filename &fileref list;
38  %put f-exist(&fileref)=%exist(fileref=&fileref);
39
40  %let library = work;
41  libname &library list;
42  %put libref(&library)=%exist(libref=&library);
43
44  %let library = workx;
45  *libname &library list;
46  %put libref(&library)=%exist(libref=&library);
47
48  %let library = library;
49  libname &library list;
50  %put libref(&library)=%exist(libref=&library);
51
52  %let fmt_lib = library.formats;
53  %put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);
54
55  %let fmt_lib = work.formats;
56  %put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);
57
58  proc format;  value x 1='one';  run;
59  %let fmt_lib = work.formats;

```

```

60 %put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);
61
62 %let macro_lib = work.sasmacr;
63 %put c-exist(&macro_lib)=%exist(catalog=&macro_lib);
64
65 %let lib_mac =
66 %sysfunc(ifc(%sysfunc(getoption(mstored)) eq MSTORED
67             ,%nrstr(%sysfunc(getoption(sasmstore)))
68             ,%nrstr(work)
69             ) );
70 %put &=lib_mac; *note .. double dots;
71 %let macro_lib = &lib_mac..sasmacr;
72 %put c-exist(&macro_lib)=%exist(catalog=&macro_lib);

```

listing exist-test.log

```

8 %put echo: %sysfunc(getoption(sasautos));
echo: (project sasautos)
11 %let data = sashelp.class;
12 %put exist(&data)=%exist(data=&data);
exist(sashelp.class)=1
13
14 %let data = work.class;
15 %put exist(&data)=%exist(data=&data);
exist(work.class)=0
16
17 %let filename = autoexec;
18 %put file-exist(&filename)=%exist(filename=&filename);
file-exist(autoexec)=0
19
20 %let filename = autoexec.sas;
21 %put file-exist(&filename)=%exist(filename=&filename);
file-exist(autoexec.sas)=1
22
23 filename autoexec 'autoexec.sas';
24 %let file_fileref = autoexec;
25 filename &file_fileref list;
NOTE: Fileref= AUTOEXEC
Physical Name= ...\\2016-SESUG-macro-exist\sas\autoexec.sas
26 %put fileref-exist(&file_fileref)=%exist(file_fileref=&file_fileref);
fileref-exist(autoexec)=1
27
28 %let fileref = work;
29 filename &fileref list;
WARNING: No logical assign for filename WORK.
30 %put f-exist(&fileref)=%exist(fileref=&fileref);
f-exist(work)=0
31
32 %let fileref = project;
33 filename &fileref list;
NOTE: Fileref= PROJECT
Physical Name= ...\\2016-SESUG-macro-exist\sas
34 %put f-exist(&fileref)=%exist(fileref=&fileref);
f-exist(project)=1
35
36 %let fileref = site_inc;
37 filename &fileref list;
NOTE: Fileref= SITE_INC
Physical Name= ...\\SAS-site\includes
38 %put f-exist(&fileref)=%exist(fileref=&fileref);
f-exist(site_inc)=1
39
40 %let library = work;
41 libname &library list;
NOTE: Libref= WORK
Physical Name= ...\\SAS Temporary Files\TD3936_DESKTOP_
42 %put libref(&library)=%exist(libref=&library);
libref(work)=1
43
44 %let library = workx;
46 %put libref(&library)=%exist(libref=&library);
libref(workx)=0
47

```

```

48 %let library = library;
49 libname &library list;
NOTE: Libref= LIBRARY
      Filename= ...\2016-SESUG-macro-exist\sas7b
50 %put libref(&library)=%exist(libref=&library);
libref(library)=1
51
52 %let fmt_lib = library.formats;
53 %put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);
c-exist(library.formats)=0
54
55 %let fmt_lib = work.formats;
56 %put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);
c-exist(work.formats)=0
57
58 proc format; value x 1='one'; run;
NOTE: Format X has been output.
NOTE: PROCEDURE FORMAT used (Total process time):
59 %let fmt_lib = work.formats;
60 %put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);
c-exist(work.formats)=1
61
62 %let macro_lib = work.sasmacr;
63 %put c-exist(&macro_lib)=%exist(catalog=&macro_lib);
c-exist(work.sasmacr)=1
64
65 %let lib_mac =
66 %sysfunc(ifc(%sysfunc(getoption(mstored)) eq MSTORED
67 ,%nrstr(%sysfunc(getoption(sasmstore)))
68 ,%nrstr(work)
69 ) );
70 %put &=lib_mac; *note .. double dots;
LIB_MAC=work
71 %let macro_lib = &lib_mac..sasmacr;
72 %put c-exist(&macro_lib)=%exist(catalog=&macro_lib);
c-exist(work.sasmacr)=1

```

macro callmacro excerpt

This macro excerpt shows the use of the functions open and attrn for two assertions: does the data set exist and does it have both n-obs and n-vars.

```

1 %MACRO callmacr
2     (data = sashelp.class /* required */
3     );
4 %*...;
5 %** description: assertions;
6 %** purpose : if fail then exit;
7 %if not(%sysfunc(exist(&data))) %then %do;
8     %put note: &sysmacroname exit not exist(&data);
9     %return;
10    %end;
11 %let dsid = %sysfunc(open (&data ));
12 %let n_obs = %sysfunc(attrn(&dsid,nobs ));
13 %let n_vars = %sysfunc(attrn(&dsid,nvars));
14 %if not &n_obs or not &n_vars %then %do;
15 %put note: &sysmacroname exit &=data &=n_obs &=n_vars;
16 %goto close_exit;
17 %end;
18 %else
19 %put note: &sysmacroname reading &=data &=n_obs &=n_vars;
20 %*...;
21 %close_exit: %let rc = %sysfunc(close(&dsid));
22 %mend callmacr;

```

Summary

Suggested Reading

- overview : Abbot [1] review side effects of use of these functions
- sysfunc : Hamilton [8], [7], Porterfield [10], and Yindra [12], review the syntax of %sysfunc and its use with data and scl functions
- exist : Hadden [5], shows how to open a data set and read its metadata, n-obs, and variable formats; Fehd [4] shows a macro that uses exist(data) and n-obs and n-vars to assert the macro will succeed
- fexist : Halpin [6], shows use of fexist in unix
- fileexist : Chaudhary and Schreiber-Gregory [2], discuss uses of data set functions for variable attributes
- getoption : Langston [9], shows the development of a macro that searches first the macro catalog, then all the folders in option sasautos for a macro
- ifc : Fehd [3], illustrates the combination of %sysfunc with ifc to add statements in open code
- libref : Richardson and Rossland [11], provide code to send a report as an e-mail attachment
- SAS OnLine : Functions and Arguments for SYSFUNC
-

Conclusion

A good tool box contains enough programs to refer to during bricolage, research, and development to assist in writing new programs that succeed with minimal testing.

Contact Information

Ronald J. Fehd Ron.Fehd.macro.maven@gmail.com

About the author: sco.wiki http://www.sascommunity.org/wiki/Ronald_J._Fehd
 LinkedIn www.linkedin.com/Ronald.Fehd
 affiliation Stakana Analytics, Senior Maverick also known as macro maven on
 SAS-L, Theoretical Programmer

Programs: http://www.sascommunity.org/wiki/
 Macro_CallMacro Macro_CallText Macro_Exist

Trademarks

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. Other brand and product names are trademarks of their respective companies.

References

- [1] David Abbott. Make SAS macros safe for others to use: Eliminate unexpected side effects. In *SouthEast SAS Users Group Conference Proceedings*, 2012. URL analytics.ncsu.edu/sesug/2012/BB-08.pdf. Beyond the Basics, 5 pp.; reviews side effects of function usage, %sysfunc: cexist, exist, open, symexist.
- [2] Kaushal Raj Chaudhary and Deanna Naomi Schreiber-Gregory. %Sysfunc is your friend. In *MidWest SAS Users Group Annual Conference Proceedings*, 2015. URL <http://www.lexjansen.com/mwsug/2015/RF/MWSUG-2015-RF-09.pdf>. Rapid Fire, 6 pp.; attrc, attrn, close, data step functions, fcmp procedure, fileexist, getoption, open, putn, %qsysfunc, varname, vartype.
- [3] Ronald J. Fehd. Using functions Sysfunc and Ifc to conditionally execute statements in open code. In *SAS Global Forum Annual Conference Proceedings*, 2009. URL <http://support.sas.com/resources/papers/proceedings09/054-2009.pdf>. Coders Corner, 10 pp.; topics: combining functions ifc, nrstr, sysfunc; assertions for testing: existence of catalog, data, file, or fileref; references.
- [4] Ronald J. Fehd. List processing macro call-macro. In *MidWest SAS Users Group Annual Conference Proceedings*, 2014. URL www.mwsug.org/proceedings/2014/BB/MWSUG-2014-BB04.pdf. Coders Corner, 19 pp.; using %sysfunc with SCL functions to read a list, a control data set, and for each observation, call a macro with variable names and values as named parameters.
- [5] Louise Hadden. Better metadata through SAS(R) II: %sysfunc, proc datasets, and dictionary tables. In *SouthEast SAS Users Group Conference Proceedings*, 2015. URL http://www.lexjansen.com/sesug/2015/57_Final_PDF.pdf. 7 pp.; attrn(dsid,nobs), close, exist, getoption, open, vformat, vtype.
- [6] Patrick Halpin. Using SAS(R) to control and automate a multi SAS program process. In *SAS Users Group International Annual Conference Proceedings*, 2006. URL www2.sas.com/proceedings/sugi31/145-31.pdf. Posters, 6 pp.; unix, done files, sleep; %sysfunc: exist, fexist.
- [7] Paul Hamilton. More fun with %Sysfunc(). In *Pacific NorthWest SAS Users Group Annual Conference Proceedings*, 1997. URL <http://www.lexjansen.com/pnwsug/1997/PNWSUG97027.pdf>. 6 pp.; read list of files in folder: dopen, dnum, dread; read file information: fopen, foptnum, foptname, finfo, fclose.
- [8] Paul Hamilton. %Sysfunc: Extending the SAS macro language. In *Pacific NorthWest SAS Users Group Annual Conference Proceedings*, 2005. URL <http://www.lexjansen.com/pnwsug/2005/how/SysFunc.pdf>. Hands On Workshop, 16 pp.; reviews syntax of %sysfunc and shows allocation of macro variables with various data and scl functions.
- [9] Rick Langston. A macro to verify a macro exists. In *SAS Global Forum Annual Conference Proceedings*, 2013. URL support.sas.com/resources/papers/proceedings13/339-2013.pdf. Quick Tips, 5 pp.; search compiled macro definitions in option sasstore and search autocall folders in option sasautos; scl: dopen(fileref), mopen(id,member-name).
- [10] Victoria Porterfield. Writing flexible SAS(R) codes: Exploring the value of global macro variables, conditional statements, and %sysfunc. In *SouthEast SAS Users Group Conference Proceedings*, 2012. URL <http://www.lexjansen.com/nesug/nesug12/cc/cc08.pdf>. Coders Corner, 10 pp.; 4 examples; attrn(dsid,nobs), close, exist(data), open.
- [11] Kari Richardson and Eric Rosslund. Using macros to automate SAS(R) processing. In *SAS Users Group International Annual Conference Proceedings*, 2004. URL www2.sas.com/proceedings/sugi29/126-29.pdf. Hands On Workshop, 15 pp.; writing macros that send e-mail with attachments; %sysfunc: exist, libref; %goto, %include.
- [12] Chris Yindra. %Sysfunc — the brave new macro world. In *SAS Users Group International Annual Conference Proceedings*, 1998. URL <http://www2.sas.com/proceedings/sugi23/Advtutor/p44.pdf>. Advanced Tutorials, 7 pp.; review of data step and scl functions available to %sysfunc, 10 examples.