PharmaSUG 2023 — Paper QT-044

Macro Code to Test Existence of Various Objects

Ronald J. Fehd, senior maverick, theoretical programmer,
Fragile-Free Software Institute

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 `filename` and `libname` 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 %sysfunc and ifc; data: `exist`, catalog: `cexist`, filename: `fileexist`, fileref of file: `fexist`, fileref of folder: `fileref`, libref: `libref`

In this paper:

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.

- `%sysfunc(fn(...))`
- table: list of verify functions
- `ifc`
- catalog
- data
- filename
- fileref of file
- fileref of folder
- libref

%sysfunc(fn(...))

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

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

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

Table 1 lists the function names and objects they verify.

Table 1 list of verify functions

object	(function	(obj.name)	)
catalog name or entry	%sysfunc(cexist	(cat-ref)	)
library member	%sysfunc(exist	(libref.memname)	)
filename.ext	%sysfunc(fileexist	(filename.ext)	)
fileref: dir+file	%sysfunc(fexist	(fileref-file)	)
fileref: directory	%sysfunc(fileref	(fileref-dir)	)
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.

syntax

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

ex.1 `True tokens` can be complete statements.

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

ex.2 `True tokens` can be tokens within a statement.

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

catalog

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

```
%let catalog = sashelp.stat;
%put
%sysfunc(ifc(%sysfunc(cexist(&catalog))
            ,%nrstr(info: exist)
            ,%nrstr(fail: not exist)
        )   ) 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.

```
%let data = sashelp.class;
%sysfunc(ifc(%sysfunc(exist(&data))
            ,%nrstr(%put info: exist data(&data);)
            ,%nrstr(%put fail: not exist data(&data);
                    endsas;) ))
```

---

filename

This example shows usage in a trivial macro function.

```
%macro exist_filename(filename);
%sysfunc(fileexist(&filename))
%mend;
%let filename = autoexec.sas;
%put
%sysfunc(ifc(%exist_filename(&filename)
            ,%nrstr(info: exist)
            ,%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.

```
filename            autoexec 'autoexec.sas';
%let fileref_file = autoexec;
%put %sysfunc(ifc(%sysfunc(fexist(&fileref_file))
             ,%nrstr(info: exist)
             ,%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.

```
%let fileref_folder = project;
%put
%sysfunc(ifc(not %sysfunc(fileref(&fileref_folder))
            ,%nrstr(info: exist)
            ,%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).

```
%put info: fmtsearch: %sysfunc(getoption(fmtsearch));
%let libref = library;
%put
%sysfunc(ifc(not %sysfunc(libref(&libref))
            ,%nrstr(info: exist)
            ,%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

Program 1 demonstrates the various exist functions.

Program 1 exist.sas

```
 /*    name: exist.sas
description: returns boolean value in (0,1)
             of existence of various objects
    purpose: centralize existence tests
       note: fileref and libref return -1,0,+1
             zero is true, non-zero is false */
%macro exist
      (data         = /*    exist */
      ,catalog      = /*   cexist */
      ,filename     = /*fileexist */
      ,file_fileref = /*   fexist */
      ,fileref      = /*   fileref */
      ,libref       = /*   libref */
      ,testing      = 0
      )/des = 'demo: exist(any object)?' ;
%let testing = %eval(not 0 eq &testing
     or %sysfunc(getoption(mprint)) eq MPRINT);
%if &testing %then %put _local_;
%if                 %length(&catalog     ) %then
         %sysfunc(   cexist(&catalog     ));
%else %if           %length(&data        ) %then
         %sysfunc(    exist(&data        ));
%else %if           %length(&filename    ) %then
         %sysfunc(fileexist(&filename    ));
%else %if           %length(&file_fileref) %then
         %sysfunc(   fexist(&file_fileref));
%else %if           %length(&fileref     ) %then %do;
      %eval(not
         %sysfunc(  fileref(&fileref)    ));
      %*zero: fileref = dir+filename.ext exist;
      %end;
%else %if           %length(&libref      ) %then %do;
      %eval(not
         %sysfunc(   libref(&libref)     ));
      %*if   libref has been assigned
             libref function returns 0
       *else returns nonzero value: -1, +1;
      %end;
%mend exist;
```

---

program exist-test.sas

Program 2 is used to test program 1.

Program 2 exist-test.sas

```
 /* name: exist-test.sas
notes:
 work.sasmacro catalog exists and contains macro exist
 autoexec.sas  allocates filenames project and site_inc
  turns on autocall:
  options mautosource sasautos = (project sasautos);
**/
%put echo: %sysfunc(getoption(sasautos));
*include project(exist)/source2;
options mprint;
%let data = sashelp.class;
%put exist(&data)=%exist(data=&data);

%let data = work.class;
%put exist(&data)=%exist(data=&data);

%let filename = autoexec;
%put file-exist(&filename)=%exist(filename=&filename);

%let filename = autoexec.sas;
%put file-exist(&filename)=%exist(filename=&filename);

filename autoexec 'autoexec.sas';
%let file_fileref = autoexec;
filename &file_fileref list;
%put fileref-exist(&file_fileref)=%exist(file_fileref=&file_fileref);

%let fileref = work;
filename &fileref list;
%put f-exist(&fileref)=%exist(fileref=&fileref);

%let fileref = project;
filename &fileref list;
%put f-exist(&fileref)=%exist(fileref=&fileref);

%let fileref = site_inc;
filename &fileref list;
%put f-exist(&fileref)=%exist(fileref=&fileref);

%let library = work;
libname &library list;
%put libref(&library)=%exist(libref=&library);

%let library = workx;
*libname &library list;
%put libref(&library)=%exist(libref=&library);

%let library = library;
```

```
libname &library list;
%put libref(&library)=%exist(libref=&library);

%let fmt_lib = library.formats;
%put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);

%let fmt_lib = work.formats;
%put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);

proc format;  value x 1='one';  run;
%let fmt_lib = work.formats;
%put c-exist(&fmt_lib)=%exist(catalog=&fmt_lib);

%let macro_lib = work.sasmacr;
%put c-exist(&macro_lib)=%exist(catalog=&macro_lib);

%let lib_mac =
%sysfunc(ifc(%sysfunc(getoption(mstored)) eq MSTORED
            ,%nrstr(%sysfunc(getoption(sasmstore)))
            ,%nrstr(work)
        )   );
%put &=lib_mac; *note    .. double dots;
%let macro_lib = &lib_mac..sasmacr;
%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
      Filename= ...\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.

```
%macro callmacr
      (data = sashelp.class /* required  */
      );
%*...;
%**  description: assertions;
%**  purpose    : if fail then exit;
%if  not(%sysfunc(exist(&data))) %then %do;
     %put note: &sysmacroname exit not exist(&data);
     %return;
     %end;
%let dsid   = %sysfunc(open (&data      ));
%let n_obs  = %sysfunc(attrn(&dsid,nobs ));
%let n_vars = %sysfunc(attrn(&dsid,nvars));
%if  not &n_obs or not &n_vars %then %do;
%put note: &sysmacroname exit &=data &=n_obs &=n_vars;
     %goto close_exit;
     %end;
%else
%put note: &sysmacroname reading &=data &=n_obs &=n_vars;
%*...;
%close_exit: %let rc = %sysfunc(close(&dsid));
%mend callmacr;
```

Summary

suggested reading

overview :	Abbott, “Make SAS Macros Safe for Others to Use: Eliminate Unexpected Side Effects” review side effects of use of these functions
sysfunc :	Hamilton, “%Sysfunc: Extending the SAS Macro Language”, Hamilton, “More Fun with %Sysfunc()”, Porterfield, “Writing Flexible SAS(R) Codes: Exploring the Value of Global Macro Variables, Conditional Statements, and %Sysfunc”, and Yindra, “%Sysfunc — The Brave New Macro World”, review the syntax of %sysfunc and its use with data and scl functions
exist :	Fehd, “List Processing Macro Call-Macro” shows macros that uses exist(data) and n-obs and n-vars to assert the macro will succeed Hadden, “Better Metadata Through SAS(R) II: %Sysfunc, Proc Datasets, and Dictionary Tables”, shows how to open a data set and read its metadata, n-obs, and variable formats; Hughes, “A Failure To EXIST: Why Testing for Data Set Existence with the EXIST Function Alone Is Inadequate for Serious Software Development in Asynchronous, Multiuser, and Parallel Processing Environments” addresses use of exist in multi-user environments
fexist :	Halpin, “Using SAS(R) to Control and Automate a Multi SAS Program Process”, shows use of fexist in unix
fileexist :	Chaudhary and Schreiber-Gregory, “%Sysfunc Is Your Friend”, discuss uses of data set functions for variable attributes
getoption :	Langston, “A Macro to Verify a Macro Exists”, shows the development of a macro that searchs first the macro catalog, then all the folders in option sasautos for a macro
ifc :	Fehd, “Using Functions SYSFUNC and IFC to Conditionally Execute Statements in Open Code”, illustrates the combination of %sysfunc with ifc to add statements in open code
libref :	Richardson and Rossland, “Using Macros to Automate SAS(R) Processing”, 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.

Author Information

Ronald J. Fehd	Ron.Fehd.macro.maven at gmail dot com
LinkedIn	https://www.linkedin.com/in/ronald-fehd-5125991/
affiliation	Senior Maverick, Theoretical Programmer, Fragile-Free Software Institute,
also known as	`macro maven` on SAS-L

Trademarks

References

Abbott, David (2012). “Make SAS Macros Safe for Others to Use: Eliminate Unexpected Side Effects”. In: *SouthEast SAS Users Group Conference Proceedings*. Beyond the Basics, 5 pp.; reviews side effects of function usage, %sysfunc: cexist, exist, open, symexist. URL: analytics.ncsu.edu/sesug/2012/BB-08.pdf.

Chaudhary, Kaushal Raj and Deanna Naomi Schreiber-Gregory (2015). “%Sysfunc Is Your Friend”. In: *MidWest SAS Users Group Annual Conference Proceedings*. Rapid Fire, 6 pp.; attrc, attrn, close, data step functions, fcmp procedure, fileexist, getoption, open, putn, %qsysfunc, varname, vartype. URL: http://www.lexjansen.com/mwsug/2015/RF/MWSUG-2015-RF-09.pdf.

Fehd, Ronald J. (2009). “Using Functions SYSFUNC and IFC to Conditionally Execute Statements in Open Code”. In: *SAS Global Forum Annual Conference Proceedings*. Coders Corner, 10 pp.; topics: combining functions ifc, nrstr, sysfunc; assertions for testing: existence of catalog, data, file, or fileref; references. URL: http://support.sas.com/resources/papers/proceedings09/054-2009.pdf.

— (2014). “List Processing Macro Call-Macro”. In: *MidWest SAS Users Group Annual Conference Proceedings*. 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. URL: http://www.mwsug.org/proceedings/2014/BB/MWSUG-2014-BB04.pdf.

Hadden, Louise (2015). “Better Metadata Through SAS(R) II: %Sysfunc, Proc Datasets, and Dictionary Tables”. In: *SouthEast SAS Users Group Conference Proceedings*. 7 pp.; attrn(dsid,nobs), close, exist, getoption, open, vformat, vtype. URL: http://www.lexjansen.com/sesug/2015/57_Final_PDF.pdf.

Halpin, Patrick (2006). “Using SAS(R) to Control and Automate a Multi SAS Program Process”. In: *SAS Users Group International Annual Conference Proceedings*. Posters, 6 pp.; unix, done files, sleep; %sysfunc: exist, fexist. URL: www2.sas.com/proceedings/sugi31/145-31.pdf.

Hamilton, Paul (1997). “More Fun with %Sysfunc()”. In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. 6 pp.; read list of files in folder: dopen, dnum, dread; read file information: fopen, foptnum, foptname, finfo, fclose. URL: http://www.lexjansen.com/pnwsug/1997/PNWSUG97027.pdf.

— (2005). “%Sysfunc: Extending the SAS Macro Language”. In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. Hands On Workship, 16 pp.; reviews syntax of %sysfunc and shows allocation of macro variables with various data and scl functions. URL: http://www.lexjansen.com/pnwsug/2005/how/SysFunc.pdf.

Hughes, Troy Martin (Oct. 2016). “A Failure To EXIST: Why Testing for Data Set Existence with the EXIST Function Alone Is Inadequate for Serious Software Development in Asynchronous, Multiuser, and Parallel Processing Environments”. In: *MidWest SAS Users Group Annual Conference Proceedings*. 8 pp.; URL: https://www.lexjansen.com/mwsug/2016/BB/MWSUG-2016-BB30.pdf.

Langston, Rick (2013). “A Macro to Verify a Macro Exists”. In: *SAS Global Forum Annual Conference Proceedings*. Quick Tips, 5 pp.; search compiled macro definitions in option sasmstore and search autocall folders in option sasautos; scl: dopen(fileref), mopen(id,member-name). URL: support.sas.com/resources/papers/proceedings13/339-2013.pdf.

Porterfield, Victoria (2012). “Writing Flexible SAS(R) Codes: Exploring the Value of Global Macro Variables, Conditional Statements, and %Sysfunc”. In: *SouthEast SAS Users Group Conference Proceedings*. Coders Corner, 10 pp.; 4 examples; attrn(dsid,nobs), close, exist(data), open. URL: http://www.lexjansen.com/nesug/nesug12/cc/cc08.pdf.

Richardson, Kari and Eric Rossland (2004). “Using Macros to Automate SAS(R) Processing”. In: *SAS Users Group International Annual Conference Proceedings*. Hands On Workshop, 15 pp.; writing macros that send e-mail with attachments; %sysfunc: exist, libref; %goto, %include. URL: www2.sas.com/proceedings/sugi29/126-29.pdf.

Yindra, Chris (1998). “%Sysfunc — The Brave New Macro World”. In: *SAS Users Group International Annual Conference Proceedings*. Advanced Tutorials, 7 pp.; review of data step and scl functions available to %sysfunc, 10 examples. URL: http://www2.sas.com/proceedings/sugi23/Advtutor/p44.pdf.