Paper 240-2011

SAS® Options - Versatile Players in the Game of SAS

Denise Poll, SAS Institute, Cary, NC

ABSTRACT

With hundreds of SAS options "in play" during a SAS session, keeping the players organized requires knowledge of:

- the position (performance, security, output control and so on) an option plays
- the timing of when the option can play (start-up, post-startup)
- the actions (get, set, reset) supported for an option

This paper explains and provides example usage of interfaces available for establishing, updating, and querying SAS options.

These are a sample of questions addressed in this paper:

- Which configuration file set an option value?
- How can an option value be reset to a previously used value, the default value, or the value established at SAS start-up?
- What is a restricted option and how can I tell if an option can be restricted or is restricted?

INTRODUCTION

SAS options provide a means for customizing a SAS Session. A SAS option consists of a unique option name and a value that is associated with the option name. SAS options are required for startup of a SAS session and are used extensively, explicitly, or implicitly throughout a SAS Session, and even during termination. The purpose of this paper is to present an overview of SAS options and provide examples that demonstrate classic features as well as enhancements for the 9.3 release of SAS.

SAS SYSTEM OPTION INTERFACES

Familiarity with the various SAS language interfaces provides a basis for evaluating the examples presented in this paper. The following is a list of interfaces with a brief explanation of each:

- **SAS Session Start-up**
  Syntax for specifying the option name and value differs by run mode (interactive, batch, and so on) and by host operating environment. For specific information, see the SAS companion documentation for your operating environment (Windows, UNIX, or z/OS).
  - **Command line** – The command that invokes SAS can contain option names and values.
  - **Administrator restricted –** Your on-site SAS support personnel can define a set of option names and values, which are established during SAS start-up and cannot be modified by other startup interfaces or post-start-up. The restricted options are usually related to performance or security.
  - **Configuration files –** One or more configuration files contain a set of option names and values. The configuration files are processed during SAS session start-up.

- **OPTIONS statement** – Use this SAS statement to set option values using syntax that identifies the option name and value.
- **PROC OPTIONS** – Use this procedure to display option properties and information about the current and default option value.
- **GETOPTION function** – Use this function in a DATA step or with the macro function %SYSFUNC. GETOPTION returns information about the properties of an option or the option value for the current value, the default value, or the value in place when the SAS session initialized. The GETOPTION function provides an interface for using the value of an option programmatically.
- **PROC SQL DICTIONARY.options** - The SAS Dictionary tables are available when using PROC SQL. SAS Dictionary tables are read-only tables (data sets) created and maintained by SAS. The OPTIONS table contains information about current SAS session options and values.
- **SASHELP.VOPTION** – VOPTION is a view into the DICTIONARY.OPTIONS table. Views are accessible from any SAS procedure or DATA step as well as the SAS Explorer window. Use this DATA step view to programmatically access information about each option. Variables are: name, type, value, description, portable/host, when the option can be used, and group name.

- **PROC OPTSAVE/DMOPTSAVE** – Saves the current SAS system option settings to the SAS registry or to a SAS data set.
- **PROC OPTLOAD/DMOPTLOAD –** Reads SAS system option settings that are stored in the SAS registry or a SAS data set and puts them into effect.
- **Options window –** An interactive window available when running the windowing environment that supports display and edit of an option. Options are organized by group and subgroup.
- **Java GetOptions and SetOptions –** The CORBA IOptionService interface supports getting and setting of SAS options. An example of this interface is not provided in this paper.
- **SCL OPTSETC, OPTGETC, OPTSETN, and OPTGETN –** Use these interfaces to get or set an option when programming SCL interfaces. An example of this interface is not provided in this paper.

WHAT OPTIONS DO I WANT TO USE?

With the hundreds of SAS options supported for all host operating environments and some specific to an individual host, how can you determine what options to customize? Fortunately, options are organized into logical groups. A SAS option can be in one or more groups. For example, the SORTSIZE option is in both the SORT and PERFORMANCE groups. The OPTIONS procedure can identify the groups and list options in each group. Also, the Options window is organized by groups and subgroups of SAS options.

To identify available groups:

```
Proc Options Listgroups; run;
```

```
Option Groups
   ….
   GROUP=COMMUNICATIONS          Networking and encryption

   Group=EMAIL                   E-mail
```

Output 1. Output from a PROC OPTIONS Procedure with the LISTGROUPS Parameter

To identify all options within a specified group:

```
Proc Options Group=(EMAIL PERFORMANCE); run;
```

```
Group=EMAIL
   …
   EMAILID=                      From E-mail address, log in ID, or profile
                                     for use with underlying e-mail system
   EMAILPORT=25                  Port number for SMTP server for e-mail
                                     method

Group=PERFORMANCE
   …
   COMPRESS=NO                   Specifies whether to compress observations
                                     in output SAS data sets
```

Output 2. Output from a PROC OPTIONS Procedure with the Group= Parameter

The display of SAS option group information using PROC SQL is shown below:

```
Proc SQL; Select * from dictionary.options Where group='PDF'; quit;
```

Option Name	Option type	Option Setting	Option Description	Option Location	Option Set	Option Group
PDFACCESS	Boolean	PDFACCESS	Allow access to PDF documents	Portable	anytime	PDF
PDFASSEMBLY	Boolean	NOPDFASSEMBLY	Allow PDF document assembly	Portable	anytime	PDF
…						

Output 3. Output from a PROC SQL Procedure

PROPERTIES OF AN OPTION

There are many properties associated with each option. Some SAS options can be specified only at start-up of a SAS session . Some can be specified at both start-up and during a SAS session. The type of an option refers to whether the option is character, numeric or Boolean. The values supported vary by type of option. For example, a character option has a property that indicates the maximum number of characters that are expected and if the option value is expected to be quoted or retained uppercased. For numeric options, minimum and maximum values are identified. When setting a numeric option, the value can be given as an explicit number such as 1024 or 1K, MIN or MAX words, or in a hexadecimal representation. For Boolean options, the value of on or off is conveyed using the syntax of NO*optname* for OFF and *optname* for ON.

The PROC OPTIONS keyword DEFINE in the following example generates the display of properties for the AUTOEXEC SAS option:

```
Proc Options Option=AUTOEXEC Define; run;
```

```
AUTOEXEC=( '!myroot\secondAuto.sas' '!myroot\firstAuto.sas' )
Option Definition Information for SAS Option AUTOEXEC
    Group= ENVFILES
    Group Description: SAS library and file location information
    Description: Identifies AUTOEXEC files used during initialization
    Type: The option value is of type CHARACTER
          Maximum Number of Characters: 1024
          Casing: The option value is retained with original casing
          Quotes: If present during "set", start and end quotes are removed
          Parentheses: The option value does not require enclosure
            within parentheses. If present, the parentheses are retained.
          Expansion: Environment variables, within the option value,
            are not expanded
    When Can Set: Environment Startup or Session Startup only
    Restricted: Your Site Administrator cannot restrict modification of this option
    Optsave: PROC Optsave or command Dmoptsave will not save this option
```

Output 4. Output from a PROC OPTIONS Procedure with the DEFINE Parameter

The following examples demonstrate ways to subset information about options and properties using PROC PRINT and PROC SQL. The *optstart* variable is available starting with SAS 9.3.

```
Proc Print Data=SASHELP.voption;
   Where optname contains 'PDF' and optstart eq 'anytime';
run;
```

Obs	optname	opttype	setting	optdesc	level	optstart	group
194	PDFACCESS	Boolean	PDFACCESS	Allow access to PDF documents	Portable	anytime	PDF
195	PDFASSEMBLY	Boolean	NOPDFASSEMBLY	Allow PDF document assembly	Portable	anytime	PDF
196	PDFCOMMENT	Boolean	NOPDFCOMMENT	Allow modification of PDF document comments	Portable	anytime	PDF

Output 5. Output from a PROC PRINT Procedure Specifying the SASHELP.VOPTION Data Set

```
Proc SQL;
   Select * from dictionary.options
      Where group eq 'PERFORMANCE' and optstart eq 'startup';
quit;
```

Option Name	Option type	Option Setting	Option Description	Option Location	Option Set	Option Group
MINPARTSIZE	num	16777216	Minimum partition size when creating SPD Engine files	Portable	startup	PERFORMANCE
SPDEMAXTHREADS	num	0	Maximum number of threads for SPD Engine processing	Portable	startup	PERFORMANCE

Output 6. Output from a PROC SQL Procedure with a Where Parameter

WHAT IS THE VALUE AND HOW DID THE VALUE COME TO BE

Each option has an initial shipped default option value. During SAS start-up, options can be set from a site restriction interface, configuration files, and the command line. After start-up, several interfaces support setting option values.

The following statement is a simple and common way to display an option value.

```
Proc Options Option=optionname; run;
```

The PROC OPTIONS keyword VALUE in the following example generates information about which interface set or augmented the SAS option value. Augmentation is achieved using the INSERT and APPEND options which are described later in this paper. The ability to identify the physical name of the configuration file that set an option value is available starting with SAS 9.3

```
Proc Options Option=AUTOEXEC Value; run;
```

```
Option Value Information For SAS Option AUTOEXEC
    Value: ('!myroot\secondAuto.sas' '!myroot\firstAuto.sas' )
    Scope: DMS Process
          How option value set: Config File
          Value Inserted: '!myroot\secondAuto.sas'
          Config file name:
            U:\config2.cfg

          How option value set: Config File
          Value Inserted: '!myroot\firstAuto.sas'
          Config file name:
            C:\SASv9\tmp\ConfigDNTNO.cfg
```

Output 7. Output from a PROC OPTIONS Procedure with a VALUE Parameter

Use a combination of the %SYSFUNC macro and the GETOPTION function to request that the option value for AUTOEXEC be displayed with the embedded environment variable resolved. The environment variable **!myroot** is expanded to the **C:\MyRoot** directory. PROC OPTIONS also supports the **EXPAND** keyword.

```
%put Expanded Autoexec Value= %sysfunc(Getoption(AUTOEXEC,EXPAND));
```

```
Expanded Autoexec Value=( 'C:\MyRoot\secondAuto.sas' 'C:\MyRoot\firstAuto.sas' )
```

Output 8. Output from the GETOPTION Function with Expansion of an Environment Variable

The following DATA step includes logic that keys off information about how the option value was set. For example, if the value for the AUTOEXEC option is set in a configuration file, the first PUT statement indicating 'Config File' is written to the SAS log.

```
Data a;
   whatval=getoption('AUTOEXEC','howset');
   If (whatval='Config File') then
      Put 'Autoexec set from a Config File';
   If (whatval='Shipped Default') then
      Put 'Autoexec not set during startup. The value is the shipped
default.';
run;
```

HOW TO QUERY OR RETURN THE DEFAULT OR STARTUP VALUE

A SAS option or options can be reset to previous settings.

Use the **STARTUPVALUE** or **DEFAULTVALUE** parameters of the GETOPTION function to generate and execute an OPTIONS statement containing either the shipped default value or the value that was set during start-up using configuration files or the command line.

The following example checks to see if the option value for YEARCUTOFF has been set during the SAS session and re-sets it to theSAS session initialized value. The *startupvalue* parameter is available starting with SAS 9.3.

```
Options YEARCUTOFF=2010;
%macro ckYear ;
%let ckval=%sysfunc(Getoption(YEARCUTOFF));
%put Current Value=&ckval;
%let startval=%sysfunc(Getoption(YEARCUTOFF,startupvalue));
%put Value at Startup=&startval;
%if (&ckval NE &startval) %then
     Options %sysfunc(Getoption(YEARCUTOFF,keyword,startupvalue));
%mend ckYear;
%ckYear;
%put Reset value=%sysfunc(Getoption (YEARCUTOFF));
```

```
Current Value= 2010
Value at Startup= 1920
76   %put Reset value= %sysfunc(getoption(YEARCUTOFF));
Reset value= 1920
```

Output 9. Output from a Macro That Uses the GETOPTION Function to Set an Option Value

SAVING AND RELOADING A SET OF SAS OPTIONS

The procedures OPTSAVE and OPTLOAD can be used to save and re-load a set of SAS options and values. SAS options that can be specified after startup and are not passwords can be saved. The set of options is saved to a SAS data set or to the SAS registry. The commands DMOPTSAVE and DMOPTLOAD also save and load option values.

The following example demonstrates how to modify a couple of SAS options, run a step and then return option values to saved settings.

```
Proc Optsave Data=WORK.saveit; run;

Options OBS=50 DATESTYLE=MDY; run;

…perform a step that uses modified option values…

Proc Optload Data=WORK.saveit; run;
```

EXPANDED INFORMATION AVAILABLE DURING STARTUP OF A SAS SESSION

Starting with SAS 9.3, specification of the option VERBOSE yields an itemized list of options for each interface or file contributing to SAS session start-up. The itemized list is written to the SAS log.

The following example shows information generated by the VERBOSE option:

```
-----------------------------------------------------------
Options set internally at initialization:
-----------------------------------------------------------
…
DBCSTYPE = WINDOWS
DMS
DMSEXP
…
```

```
-----------------------------------------------------------------
Options specified in the config file C:\SASv9\tmp\SASv9-356.cfg:
-----------------------------------------------------------------
SET = SASROOT C:\SASv9
…
SORTSIZE = 2m
CATCACHE = 0
…
```

Output 10. Output from the VERBOSE Option

SAS OPTIONS INSERT AND APPEND APPLY TO A FEW OPTIONS

INSERT and APPEND options support the ability to add to the beginning or end of an option value. Starting with SAS 9.3, the ability to identify which SAS options are impacted by INSERT and APPEND is supported. In addition, specific information about how INSERT and APPEND contribute to the construction of a SAS option value is also available.

Use **PROC OPTIONS LISTINSERTAPPEND;** to determine which SAS options allow the option value to be modified by INSERT and APPEND.

```
Options INSERT=(FMTSEARCH=MYLIB); run;
Proc Options Option=FMTSEARCH VALUE; run;
```

The following identifies the full option value and how each portion of the option value was set.

```
Proc Options option=FMTSEARCH VALUE ;

Option Value Information For SAS Option FMTSEARCH
    Value: (MYLIB WORK LIBRARY)
    Scope: DMS Process
          How option value set: Options Statement
          Value Inserted: MYLIB

          How option value set: Shipped Default
          Value: WORK LIBRARY
```

Output 11. Output of an Option Value Impacted by the INSERT= Option

WHAT IS A RESTRICTED OPTION?

Restricted options are SAS options whose values are established by on-site SAS Support personnel and cannot be overridden during or after start-up of a SAS session. Many SAS options can be restricted. Restriction of one or more SAS options is not required. Performance or security related SAS options are most often selected for restriction. During start-up, if an attempt to set a restricted option is made, the set request is ignored. After start- up, if an attempt to set a restricted option is made, a warning message is written to the SAS log.

Use **PROC OPTIONS LISTRESTRICT; RUN;** to determine which SAS options can be restricted.

Use **PROC OPTIONS RESTRICT; RUN;** to determine which SAS options are restricted.

This output in the SAS log shows that the **CMPOPT** option is restricted:

```
Option Value Information For SAS Option CMPOPT
    Option Value: (NOEXTRAMATH…..)
    Option Scope: SAS Session
    How option value set:  Site Administrator Restricted
```

Output 12. Output for an option that is restricted

This message from the SAS log when running **PROC OPTIONS RESTRICT; RUN;** shows that no options are restricted.

```
Your site administrator has not restricted any options.
```

Output 13. Output when no options are restricted

CONCLUSION

The existing SAS option interfaces and new features added for SAS 9.3 provide a versatile set of tools for controlling and reflecting the status of a SAS session. Examples demonstrate enhancements to information available during start-up of a SAS session, the OPTIONS procedure, and how to programmatically use the GETOPTION function to support your computing needs.

REFERENCES

SAS Institute, Inc. 2009. *SAS® 9.2 Language Reference: Dictionary, Third Edition*. Cary, NC: SAS Institute, Inc. Available at http://support.sas.com/documentation

SAS Institute, Inc. 2009. *SAS® 9.2 Companion for UNIX Environments.* Cary, NC: SAS Institute, Inc. Available at http://support.sas.com/documentation/cdl/en/hostunx/61879/PDF/default/hostunx.pdf

SAS Institute, Inc. 2009. *SAS® 9.2 Companion for Windows*, Second Edition Cary, NC: SAS Institute, Inc. Available at http://support.sas.com/documentation/cdl/en/hostwin/63285/PDF/default/hostwin.pdf

SAS Institute, Inc. 2009. *SAS® 9.2 Companion for z/OS* Cary, NC: SAS Institute, Inc. Available at http://support.sas.com/documentation/cdl/en/hosto390/61886/PDF/default/hosto390.pdf

ACKNOWLEDGMENTS

I would like to thank Gloria Cappy for reviewing this paper. A tip of the hat goes to Elizabeth Maldonado for editing this paper and managing SAS Options related documentation. In addition, recognition goes out to the following developers and testers who contribute to the SAS options experience: David Aronhalt, Gloria Cappy, Christy Carter, Kent Fiala, Eric Horton, Marc Howell, David Poppe, and John Turnley.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Denise Poll
Enterprise: SAS Institute Inc.
Address: SAS Campus Drive
City, State ZIP: Cary, NC 27513-2414
E-mail: Denise.Poll@sas.com