Removing Macro Variables from the SAS® Environment

Frank C. DiIorio, Advanced Integrated Manufacturing Solutions, Co.
Durham, North Carolina

THE PROBLEM

A common, and sometimes irksome, feature of running SAS programs interactively (in Display Manager or via SAS/Intrnets's socket services) is carryover of values. Titles, footnotes, macro variable values and options that are set in submission "x" are still in effect in submission "x+1" even if "x+1" did not explicitly reset the values.

Sometimes this is desirable, sometimes it is not. Regardless, it is the programmer's responsibility to control these settings. Some are easy to control (titles, footnotes, system options). But what about macro variables? There are numerous techniques for nulling the value of the variable [%let var = %str(); is a favorite] but this does not actually *remove* the variable from the macro symbol table.

A SOLUTION

This paper demonstrates a method for *removing*, rather than *nulling*, a macro variable from a SAS session. The key is to force SAS to write the macro variable to a catalog, rather than to memory. Once in the catalog, the entry containing the variable definition may be deleted like any other entry.

The paper:

- identifies the system option needed to force variables to catalog entries
- shows SAS's macro variable management behavior before and after invocation of the system option
- identifies the naming convention for the macro variable catalog
- shows how to configure SAS so that macro variables are always "delete-able"
- presents a macro that can be used to automate macro variable deletion

Note that this paper is based on code written and tested in SAS System Version 7.00, System Level TS 00P1 running under Windows 95. Similar results in other SAS releases and operating systems are *implied* but, of course, not guaranteed.

THE MVARSIZE SYSTEM OPTION

The SAS System help file says it best, and gives a hint as to the solution of the variable removal problem.

> The MVARSIZE system option specifies the maximum size for macro variables that are stored in memory. *If the size of the macro variable is larger than the maximum value that is specified, variables are written out to disk* [emphasis added].

Macro variable definitions are typically stored in memory in an area whose size is defined by MVARSIZE. If the program needs more space to store the macro variable than is available in memory, the program does *not* fail. Instead, SAS writes the macro to temporary disk storage. It does so without issuing a message (Note or Warning) to the SAS Log.

It stands to reason that if the value of MVARSIZE is set to 0, SAS will write all macro variables to disk. This will, in turn, enable all macro variables to be deleted from disk. The value may be reset from an OPTIONS statement, during SAS initialization (batch, Web-based, or interactive), or from the OPTIONS window or statement. The following OPTIONS statement is valid:

```
options mvarsize=0;
```

A Note About MSYMTABMAX. Macro aficionados may note that the MYSYMTABMAX option achieves the same result as MVARSIZE. MSYMTABMAX identifies the maximum amount of memory that is available to macro variable symbol tables. Like MVARSIZE, once the limit is reached, SAS writes variable definitions to disk. The impact of either option is, given limited experimentation while preparing this paper, identical.

THE EFFECT OF USING MVARSIZE

Suppose a SAS session began with the usual default value of MVARSIZE (usually 4,096 bytes). The statement

```
%let x = %str(pre-MVARSIZE);
```

would place macro variable X in memory with a value of "pre-MVARSIZE". Now we submit an OPTIONS statement similar to the one above and define a new macro variable;

```
options mvarsize=0;
%let y = %str(post-MVARSIZE);
```

We now have two macro variables. X is still in memory. When SAS attempted to write Y, however, it compared the maximum length for macro variables (0 bytes) to the length of variable Y (13). Since there was insufficient available memory, the macro variable was written as a member of a temporary catalog (WORK.SAS0ST0 [those are zeros]). The catalog name varies, depending on the host operating system and release of the SAS System. The complete reference to Y is:

```
work.sas0st0.y.msymtab
```

Variable Y may be used *exactly* like X or any other macro variable. Its distinguishing characteristic is that it is resident on disk rather than memory.

Now suppose we change variable X:

```
%let x = %str(new value);
```

X was memory resident but now there is no memory available for storing macro variables. SAS behaves exactly the same way as when it wrote variable Y. We now have another member of catalog SAS0ST0:

```
work.sas0st0.x.msymtab
```

And macro variable X, like Y, may now be deleted from the SAS session.

DELETING MACRO VARIABLES

Once the macro variable is stored as a catalog entry, it is a pretty straightforward matter to delete it. The following statements delete the catalog entry containing the definition of macro variable X:

```
proc catalog c=work.sas0st0;
delete x / et=msymtab;
quit;
```

Multiple variables may be deleted by indicating multiple entry (i.e., variable) names in the DELETE statement:

```
proc catalog c=work.sas0st0;
```

```
delete x y / et=msymtab;
quit;
```

Use the following syntax to delete all members of the catalog:

```
proc catalog c=work.sas0st0 kill;
quit;
```

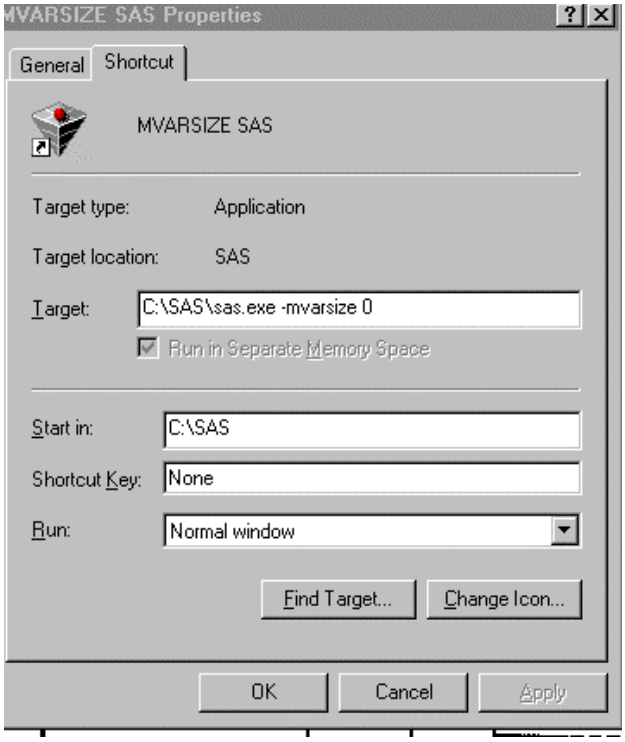
A macro to simplify and "bulletproof" this technique is presented in Appendix A.

AUTOMATICALLY ENABLING VARIABLE DELETION

Manually downsizing the value of MVARSIZE is fine, but what if you want your application to *always* store macro variable definitions on disk, thus enabling variable deletion? As is always the case with robust tools such as the SAS System, there are many ways to do this. This section identifies a few of them.

Icon Setting MVARSIZE

In graphical environments such as Windows 95/98/NT, OpenView, *et al.*, the user can define the command string that is executed when an icon is activated. This allows the user to tell SAS what options, startup files, and the like to use during startup. To set the MVARSIZE option in a Windows NT icon definition screen, enter text as follows:

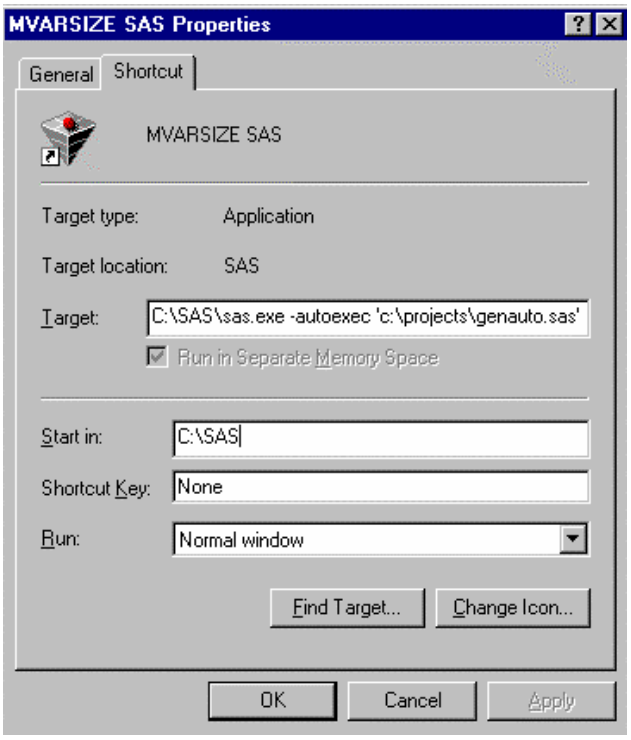


Icon Pointing to an AUTOEXEC File

Another option in either the Startup icon definition or as part of a batch file or script is the setting MVARSIZE in an "autoexec" file. The autoexec file contains SAS statements that are executed before the program file begins or control is turned over to the user in a Display Manager session. Autoexec file contents include, but are not limited to: setting options, defining default titles, and allocating libraries and external files. Suppose the contents of file c:\projects\genauto.sas is:

```
libname prod 'c:\proddata';
options mvarsize=0;
```

The SAS startup icon could be defined as follows:



The autoexec file could also be pointed to by a script (BAT file, EXEC, .KSH file, etc.) that initiates a SAS session.

Web-Based Applications

There are a number of ways to set the MVARSIZE (and other) options during startup of Web-based sessions. One approach for launch services is modification of SRVAUTO.SAS. The file is typically located in the folder identified by the SRVROOT option for the service. The service is located in the SAS/Intrnet BROKER.CFG (broker configuration) file. Restated:

- BROKER.CFG contains service definitions, and
- the service definition contains an INITCMD option, which specifies the SRVROOT folder, and
- SRVAUTO.SAS resides in the SRVROOT folder and contains the MVARSIZE option

Not obvious at first blush, but something Web and CGI programmers get used to.

Socket services use the file RESET.SAS (usually resident in the SAS/Intrnet directory) to set options such as MVARSIZE. Quoting from the header comment in the SAS Institute-supplied RESET.SAS file:

```
This file contains code that gets executed
prior to each program that the application
server runs.  Include [any] SAS statements
here which you require to ensure that the
server session is properly reset.
```

The sample file contains null titles and footnotes, options, and goptions.

Removing Macro Variables from the SAS Environment

SUMMARY

Deleting macro variables from a SAS session is both possible *and* straightforward once one is aware of the proper use of MVARSIZE system option. The performance hit incurred by the additional disk I/O is more than offset by the improved reliability of the program's performance. The techniques for option setting and macro variable deletion suggest there is a variety of methods for implementation.

QUESTIONS? COMMENTS?

Frank Dilorio 919.942.2028
Advanced Integrated fcd1@mindspring.com
Manufacturing Solutions, Co.
102 Westbury Drive
Chapel Hill, NC 27516-9154

APPENDIX A: A MACRO TO DELETE MACRO VARIABLES

The following macro deletes one or more macro variables and creates a macro variable indicating the status of the deletion procedure. It assumes the macro variable(s) to delete are in the SAS0TS0 catalog (i.e., the MVARSIZE option has been set to 0 prior to macro invocation and variable assignment). Refer to the header comment for usage notes.

```
/*

DELMVAR - delete macro variable

Parameters (both are keyword parms and case
     insensitive)
=> macrovar - name of macro variable to delete
            - no default
              [all] deletes all variables
                    stored in the temporary
                    (SAS0TS0) catalog
=> status   - status/return code variable
            - default is _STATUS_
            - returned values
              NS    Parm MACROVAR not specified
              NDEL Macro variable(s) could not
                   be deleted
              DEL  Macro variable(s) deleted
                   successfully

*/

%macro delmvar(macrovar=, status=_status_);
%put DELMVAR-> Begin. Delete &macrovar  Return
               status var &status;

%let macrovar=%upcase(&macrovar);
%let status  =%upcase(&status);

* GLOBALize status var and set default value ;
%global &status;
%let &status = NF;

%if "&macrovar" ne "" %then %do;
    %put DELMVAR-> Non-blank MACROVAR
                   parameter;

    %let &status=NDEL;
    %local _nbefore _after _globals;

    * See if we can locate the catalog and,
      maybe, the variable ;
    proc sql noprint;
    select count(*) into :_nbefore
    from dictionary.catalogs
    where libname = "WORK"      &
          memname = "SAS0ST0"   &
          objtype = "MSYMTAB"   &
          objname = "&macrovar" ;
    %end;
    %else %do;
          objtype = "MSYMTAB" ;
          %end;
    quit;

    %if &_nbefore. ne 0 %then %do;
        %put DELMVAR-> Found catalog-entry;

        %if &macrovar = [ALL] %then %do;
            proc sql noprint;
            select name into :_globals
                   separated by " "
            from dictionary.macros
            where scope = "GLOBAL" and name ne
                  "&status";
            quit;
            %end;

        * Now try to delete ;
        proc catalog c=work.sas0st0
             entrytype=msymtab;
        delete
        %if &macrovar ne [ALL] %then
            &macrovar. %str(;) ;
            %else &_globals. %str(;) ;
        quit;

        * See if we have successfully deleted.
          The only global var left should be
          the status variable. ;
        proc sql noprint;
        %if &macrovar = [ALL] %then %do;
            select name into :_after separated
                   by " "
            from dictionary.macros
            where scope = "GLOBAL";
            %end;
            %else %do;
                  select count(*) into :_after
                  from dictionary.catalogs
                  where libname= "WORK"      &
                        memname= "SAS0ST0"   &
                        objtype= "MSYMTAB"   &
                        objname= "&macrovar" ;
                  %end;
        quit;
        %if &macrovar = [ALL] and
            "&_after" = "&status"
            %then %let &status=DEL;
            %else %if &_after = 0
                  %then %let &status=DEL;
        %end;

    %end;

%put DELMVAR-> Status var. &status=&&&status;
%mend;
```