

Using a Control Dataset to Manage Production Compiled Macro Library

Curtis E. Reid, Bureau of Labor Statistics, Washington, DC

ABSTRACT

By default, SAS® compiles and stores all macros into the WORK library. This creates an overhead when a programmer executes a SAS® program. By compiling and saving these macros into a permanent library, the SAS® program can save compilation time. The drawback is that many macros often need to be updated and recompiled as specifications or requirements change. Additionally, many organizations now require that their programs have version control or software configuration management process in place. Thus updating a macro(s) in a SAS® program becomes a non-trivial, time-consuming task. The use of a make-compile-control-dataset removes the need to manually compile all macros by reading in the make-compile-control-dataset, obtaining correct macro(s) to compile, and storing into the library. The value added from using this programming tool is that it facilitates maintenance and reduces business risks.

INTRODUCTION

This paper discusses the concept of using a make-compile-control-dataset (MCD) to manage compilation of macros in production systems. A production system is a series of interrelated administrative and analytical datasets, programs, and end-user tools that deliver mission-critical information (such as statistical analyses, database extracts, and management reports). The MCD is a SAS® dataset consisting of four basic variables. This dataset is a non-code dependent resource that determines what macros need to be compiled or not.

PROBLEM

Typically, a production system is one that is inherently stable; changes to the programs are often discouraged to ensure reliability. In organizations with a software configuration management process in place, updating a compiled macro library requires that: (1) a change request comes in and receives approval; (2) the approved change gets assigned to the programmer; (3) the programmer checks out the program, modifies the macro(s) and checks the code back in; (4) the revised macro is tested and approved; and (5) then the updated code is compiled into the compiled macro library. In order to compile a macro, a few lines of code must be inserted at the beginning of the macro file. These few lines of code are redundant and in some case it can generate an error message complaining that the LIBNAME reference has already been defined.

SOLUTION

The use of an MCD minimizes the need to modify any new or existing macros with extra lines of codes for compilation purpose. The MCD will contain all of the necessary information such as the pathname where the macro file is located, the filename, an identifier indicating whether it is to be compiled or not, and the date of last time a macro was compiled. A calling program executes an included macro to read variables in from the control dataset and generate global macro variables. In turn, these global macro variables will be used by the calling program which will perform a few steps and include the macro to be compiled into the compiled macro library. Any changes in the MCD are transparent to the calling program.

A WALK-THROUGH APPROACH

For the purpose of this paper, the process flow and the sample program are simplified to demonstrate the concept of using control data into production programs.

The main or calling program sets up the necessary LIBNAME references and includes a couple macros. Then, it calls a main macro %DO_COMPILE which will be explained in detail shortly. After the %DO_COMPILE has finished, it checks for the condition of the program. If it compiled successfully, then an email is sent out indicating a success; else an email is sent out indicating a failure.

In the %DO_COMPILE macro, a few processes are involved. Two parameters are used which are the default: INITIAL_YN and INITIAL_PATH. These two parameters apply only for the first time like most of you will be doing.

The only cases where INITIAL_YN = 'YES' apply to a special macro called %COMPSTOR. In case you didn't know about this macro, it will compile the following SAS-supplied macros into the compiled macro library. These SAS-supplied macros are: %CMPRESS, %DATATYP, %LEFT, %QCOMPRESS, %QLEFT, %QTRIM, %TRIM and

%VERIFY. Once these SAS-supplied macros are compiled, it is no longer necessary to recompile in the future unless there is a version change (e.g., SAS v8 to v9). The second parameter INITIAL_PATH is used for the %COMPSTOR macro to point to the development compiled macro library.

For the most part, INITIAL_YN will be 'NO'. In this case, it will delete a working copy in the development directory so that it could not accidentally be used again. Next, backup the production library to the archive library and then copy the production library to development. In the archive library, rename the library with the date of the run's date.

Now, the rest of the program will execute regardless of the value of INITIAL_YN or INITIAL_PATH. First, find out how many observations there are in the make-compile-control-dataset or MCD. Then, use that number to perform a DO loop. For each observation read in, generate Macro Variables containing the values of PATH, PROGRAM and MAKE. For every instance where the value of MAKE equals 'Y', check to make sure the actual file exists and if so, include the source file of the macro to be compiled. If an error occurred at this point, we jump to the label ENDLOOP to exit and return back to the calling program. (Please bear in mind that in order for the macro to be compiled, the %MACRO statement must have an additional option 'STORE'. Without 'STORE', SAS® will treat it as if it is an included macro.)

Once the source file has been included, then the program will check the value of &SYSCC to determine whether the included macro was successful or not. If an error has occurred, jump to the label ENDLOOP to exit and return back to the calling program. It keeps track what was included by creating a dataset called COMPILED which reads in a specific observation from the MCD and set the value of MAKE to N along with the run date. By setting the value of MAKE = 'N', this ensures that this source file cannot be compiled again unless the user explicitly want to do it again via MCD.

Now the dataset COMPILED has been created. If there is a problem with the source file and it could not be included, then COMPILED dataset would not create properly which is where we can capture this error via &SYSERR. If an error occurred then it jumps to the label %ENDLOOP. This program knows that the dataset UPDATED will not be created so it will simply skip the remainder of the program and return back to the calling program.

Once COMPILED dataset was created successfully, then this information need to add to the dataset UPDATED. As long as UPDATED has observations, produce a listing report documenting what source macro files were updated during this program run.

Now, the program will update a working copy of MCD with values from UPDATED. Once that is done, then it deletes the production version of MCD and makes a working copy of MCD a permanent production version. Next, delete the permanent production compiled library and copy the development compiled library over to the production library.

A supporting macro %NOBS is used. This macro generates a macro variable &NOBS by reading in the number of observation from a dataset. It has been used around for ages. I am quite sure most of you would say use PROC SQL or some other method to obtain a value. This works very well under any conditions.

SAMPLE PROGRAM

MAIN PROGRAM

```
OPTIONS COMPRESS = YES REUSE = YES MPRINT NOOVP MAUTOSOURCE;
/*****
/*
/* PROGRAM: make_compile.sas
/* PURPOSE: THIS PROGRAM READS IN FROM THE CONTROL DATASET AND FINDS
/* RECORDS THAT NEEDS TO BE COMPILED.
/*
/* AUTHOR: CURTIS E. REID
/* DATE: FEBRUARY 13, 2006
/*
/* MODIFICATION HISTORY
/* DATE AUTHOR DESCRIPTION
/*
*****/
%GLOBAL WTOPDIRECTORY WMACDEV NAME GOODMSG ERRORMSG MESSAGE;
```

```

%MACRO MAKE_COMPILE;

LIBNAME MACDEV "&WTOPDIRECTORY.&WMACLIB./development";
LIBNAME MACARC "&WTOPDIRECTORY.&WMACLIB./archive";
LIBNAME MACPROD "&WTOPDIRECTORY.&WMACLIB." ACCESS = READONLY;
LIBNAME MACUPD "&WTOPDIRECTORY.&WMACLIB.";
LIBNAME DATALIB "&WTOPDIRECTORY./data";

/* WE MUST USE THE FOLLOWING INCLUDE STATEMENT BECAUSE WE ARE NOT */
/* YET CALLING THE COMPILED LIBRARY UNTIL WE HAVE FINISHED COPYING */
/* THE COMPILED LIBRARY TO ARCHIVE AND DEVELOPMENT DIRECTORIES. */
%INCLUDE "&WTOPDIRECTORY./programs/do_compile.sas";
%INCLUDE "&WTOPDIRECTORY./programs/nobs.sas";

%DO_COMPILE;

%IF (&ERRCOND = 0) %THEN %DO;

FILENAME OUTBOX EMAIL TO = "&NAME"
SUBJECT = "&GOODMSG";

DATA _NULL_;
FILE OUTBOX;

PUT 'A new version of compiled Macro Library is saved into';
PUT "&WTOPDIRECTORY.&WMACLIB..";
RUN;

%END; /* GOOD RUN */
%ELSE %DO;

/* I HOPE THIS WORKS WHEN SAS ENCOUNTERS A FATAL ERROR THAT SETS */
/* OBS = 0. THE FOLLOWING OPTIONS TURN IT BACK ON AND INFORM */
/* THE USER OF A PROBLEM. IT WORKS OCCASSIONALLY BUT NEEDS TO BE */
/* REFINED BETTER... */
OPTIONS OBS = MAX REPLACE NO$SYNTAXCHECK;

FILENAME OUTBOX EMAIL TO = "&NAME"
SUBJECT = "&ERRORMSG";

DATA _NULL_;
FILE OUTBOX;

PUT 'A new version of compiled Macro Library is NOT saved into';
PUT "&WTOPDIRECTORY.&WMACLIB..";
PUT "Review the log in &WTOPDIRECTORY./programs/make_compile.log.";
PUT "ERROR MESSAGE: &MESSAGE";
RUN;

%END; /* NOT GOOD RUN */
%MEND MAKE_COMPILE;

/*****
/*
/* BEGIN MAIN PROCESSING
/*
/*
*****/
%LET WTOPDIRECTORY = /home/reidc/blssasug;
%LET WMACLIB = /macros;
%LET NAME = %STR(reid.curtis@bls.gov);

```

```
%LET GOODMSG      = %STR(SUCCESS! Compiled Macro Library Updated);
%LET ERRORMSG     = %STR(FAILURE! Compiled Macro Library NOT Updated);
```

```
%MAKE_COMPILE
```

``` %DO_COMPILE MACRO ```

```

/*****
/*
/* PROGRAM: DO_COMPILE
/* PURPOSE: This program will read in from a control dataset and compile
/* macros that are marked with a value of 'Y' in MAKE. The
/* control dataset is manually updated by changing any existing
/* macros in MAKE to 'Y' or add new records with a value in
/* MAKE = 'Y'. Once the macros have been compiled, this
/* program will produce a listing of macros compiled in this
/* run and change the value in MAKE to 'N'. Additionally,
/* before the compiled macro is run, the original macro is
/* copied to both archive and development directories. When
/* this process is finished, the newly compiled macro library
/* (saved in another directory) is copied back to the original
/* (production) directory to be used immediately.
/*
/* AUTHOR: CURTIS E. REID
/* DATE: FEBRUARY 13, 2006
/*
/* MODIFICATION HISTORY:
/* DATE AUTHOR DESCRIPTION
/*
*****/

```

```
%MACRO DO_COMPILE(INITIAL_YN = NO,
                  INITIAL_PATH = %STR(/development));
```

```

/* SET UP INITIAL GLOBAL AND LOCAL MACRO VARIABLES */
%GLOBAL ERRCOND;
%LOCAL I TTL;
%LET ERRCOND = ;
%LET TTL = 0;

```

```

/* IF THIS IS AN INITIAL RUN, DO NOT PERFORM A BACKUP */
/* ELSE PERFORM A BACKUP BEFORE COMPILING ANYTHING. */
%IF (%UPCASE(&INITIAL_YN) = NO) %THEN %DO;

```

```

/* DELETE EXISTING VERSION IN DEVELOPMENT */
PROC DATASETS LIB = MACDEV
    MEMTYPE = CATALOG NOWARN NOLIST;
    DELETE SASMACR;
RUN;
QUIT;

```

```

/* CREATE A BACKUP FROM PRODUCTION TO ARCHIVE */
/* COPY PRODUCTION TO DEVELOPMENT TO COMPILE */
PROC DATASETS LIB = MACPROD
    MEMTYPE = CATALOG NOLIST;
    COPY OUT = MACARC;
    SELECT SASMACR;
    COPY OUT = MACDEV;
    SELECT SASMACR;
RUN;
QUIT;

```

```

/* DELETE SAME-DAY ARCHIVE COPY, IF ANY */

```

```

/* APPEND DATE STAMP TO THE ARCHIVE */
PROC DATASETS LIB = MACARC
    MEMTYPE = CATALOG NOWARN NOLIST;
    DELETE SASMACR_&SYSDATE9.;
    CHANGE SASMACR = SASMACR_&SYSDATE9.;
RUN;
QUIT;

%END; /* NOT INITIAL PROCESS */

/* ONLY PROCESS THE FOLLOWING IF THIS IS AN INITIAL PROCESS */
/* TO CREATE A NEW COMPILED LIBRARY */
%IF (%UPCASE(&INITIAL_YN) = YES) %THEN %DO;

    /* COMPILE AUTOCALL MACROS */
    /* %CMPRES          %QLEFT */
    /* %DATATYP        %QTRIM */
    /* %LEFT           %TRIM  */
    /* %QCMPRES        %VERIFY */
    %COMPSTOR(PATHNAME = %STR(&WTOPDDIRECTORY.&WMACLIB.&INITIAL_PATH.));

%END;

/* NOW WE ARE FINISHED WITH COPYING OF THE COMPILED */
/* LIBRARY, THEN TELL SAS TO BEGIN USING THE COMPILED */
/* LIBRARY IN DEVELOPMENT FOR PROCESSING. */
OPTIONS MSTORED SASMSTORE = MACDEV;

/* FIND OUT HOW MANY MACRO PROGRAMS FROM THE CONTROL */
/* DATASET AND THEN LOOP THROUGH EACH OBSERVATION AT */
/* A TIME. */
%NOBS(DATA = DATALIB.MAKE_COMPILE)
%LET TTL = &NOBS;

%DO I = 1 %TO &TTL;

    DATA _NULL_;
        SET DATALIB.MAKE_COMPILE;

        IF (_N_ = %EVAL(&I)) THEN DO;

            /* PUT THE VALUES FROM THE SAS DATASET INTO */
            /* SAS MACRO VARIABLES */
            CALL SYMPUT('PATH', PATH);
            CALL SYMPUT('PROGRAM', PROGRAM);
            CALL SYMPUT('MAKE', MAKE);

            END; /* _N_ = &I */
        RUN;

        /* WE NEED TO COMPILE THOSE PROGRAMS IF THE VALUE */
        /* OF MAKE EQUALS 'Y' (YES). ELSE, DON'T TRY TO */
        /* COMPILE THAT DOESN'T NEED COMPILING. */
        %IF (%UPCASE(&MAKE) EQ Y) %THEN %DO;

            /* TURN ON OPTION SOURCE2 TO SHOW WHAT HAS BEEN */
            /* INCLUDED IN COMPILING INTO THE COMPILED MACRO */
            /* LIBRARY */
            OPTION SOURCE2;

            /* CHECK FOR THE EXISTENCE OF THE INCLUDE MACRO FILE */

```

```

%IF (%SYSFUNC(FILEEXIST(%CMPRES(&PATH)/%CMPRES(&PROGRAM))) = 0) %THEN %DO;

    %LET ERRCOND = 1; /* ASSIGN A VALUE OTHER THAN ZERO */
    /* SET AN ERROR MESSAGE FOR THE EMAIL */
    %LET MESSAGE = THE FILE DOES NOT EXIST;
    /* END THIS LOOP */
    %GOTO ENDLOOP;

%END;

/* RESET THE VALUE OF SYCC TO ZERO AT EACH ITERATION */
%LET SYSCC = 0;

/* THIS STATEMENT WILL CAUSE THE MACRO TO BE STORED */
/* INTO THE COMPILED MACRO LIBRARY AS LONG THE */
/* MACRO STATEMENT CONTAINS THE STORE OPTION. */
%INCLUDE "%CMPRES(&PATH)/%CMPRES(&PROGRAM)";

/* FIND OUT IF THE INCLUDED FILE WAS A SUCCESS OR NOT */
/* A VALUE OF ZERO OR LESS THAN 4 IS A GOOD RUN */
/* ANY VALUE GREATER THAN 4 INDICATES AN ERROR */
%LET ERRCOND = &SYSCC;
%IF (&ERRCOND > 4) %THEN %DO;

    /* SET AN ERROR MESSAGE FOR THE EMAIL */
    %LET MESSAGE = A PROBLEM ENCOUNTERED DURING INCLUDE OF MACRO FILE;
    /* END THIS LOOP */
    %GOTO ENDLOOP;

%END;

/* KEEP A TRACK WHAT HAS BEEN COMPILED BY MATCHING */
/* UP WITH THE OBSERVATION NUMBER AND SET THE VALUE */
/* IN MAKE TO 'N' (NO) SO NO FURTHER COMPILEATION */
/* WILL BE NECESSARY. */
DATA COMPILED;
    ATTRIB DTE_UPD FORMAT = DATE9.
           LABEL = 'Date Compiled';

    SET DATALIB.MAKE_COMPILE;

    IF (_N_ = %EVAL(&I)) THEN DO;

        MAKE = 'N';
        DTE_UPD = TODAY();

        OUTPUT COMPILED;

    END; /* _N_ = &I */
RUN;

/* MAKE SURE THAT WE DIDN'T ENCOUNTER AN ERROR IN */
/* THE SAS UPDATE DATASET COMPILED. */
%LET ERRCOND = &SYSERR;
%IF (&ERRCOND = 0) %THEN %DO;

    /* IF THIS IS NOT THE FIRST TIME WE'VE COMPILED */
    /* MACROS IN THIS COMPILED LIBRARY, SIMPLY DO */
    /* AN APPEND UPDATED WITH COMPILED. */
    /* */
    /* IF THIS IS THE FIRST TIME, THEN CREATE A COPY*/

```

```

/* OF COMPILED AS UPDATED.                                     */
%IF (%EVAL(&I) > 1) %THEN %DO;

    PROC APPEND BASE = UPDATED
              DATA = COMPILED;
    RUN;

%END;
%ELSE %DO;

    DATA UPDATED;
    SET COMPILED;

    OUTPUT UPDATED;
    RUN;

%END; /* IF &I > 1 */
%END;
%ELSE %DO;

    /* NOT GOOD SO END THIS LOOP */
    %GOTO ENDLOOP;

%END; /* ERROR */
%END; /* MAKE = YES */
%ELSE %DO;

    /* SET AN ERROR MESSAGE FOR THE EMAIL */
    %LET MESSAGE = UNABLE TO CREATE DATASET COMPILED;

%END; /* PROBLEM ENCOUNTERED */
%END; /* DO I = 1 TO &TTL LOOP */

%ENDLOOP:

/* IF NO ERROR WAS ENCOUNTERED DURING THIS COMPILATION, */
/* CONTINUE WITH THE REMAINDER OF THE MACRO PROGRAM. */
/* ELSE IF AN ERROR OCCURRED, THIS SECTION IS NOT RUN */
/* WILL GO BACK TO THE MAIN PROGRAM TO GENERATE AN EMAIL */
%IF (&ERRCOND = 0) %THEN %DO;

    /* FIND OUT HOW MANY ARE IN THE UPDATED DATASET */
    %NOBS(DATA = UPDATED)

    /* IF &NOBS IS GREATER THAN ZERO, THIS IS A GOOD RUN. */
    /* IF &NOBS IS ZERO OR NEGATIVE, THIS IS A BAD RUN. */
    %IF (&NOBS > 0) %THEN %DO;

        /* SHOW WHAT HAS BEEN COMPILED IN THIS RUN */
        PROC PRINT DATA = UPDATED LABEL;
        VAR PATH PROGRAM DTE_UPD;
        TITLE 'MACROS COMPILED IN THIS RUN';
        RUN;

        /* SORT BOTH MAKE_COMPILE AND UPDATED */
        /* WHICH IS REQUIRED PRIOR TO UPDATING */
        /* THE BASE DATASET MAKE_COMPILE */
        PROC SORT DATA = DATALIB.MAKE_COMPILE;
        BY PROGRAM;
        RUN;

```

```

PROC SORT DATA = UPDATED;
  BY PROGRAM;
RUN;

/* UPDATE THE BASE DATASET MAKE_COMPILE INTO */
/* WORK LIBRARY */
DATA MAKE_COMPILE;
  UPDATE DATALIB.MAKE_COMPILE
    UPDATED;
  BY PROGRAM;

  OUTPUT MAKE_COMPILE;
RUN;

/* RE-SORT THE UPDATED MAKE_COMPILE BEFORE */
/* WRITING TO PERMANENT SAS LIBRARY */
PROC SORT DATA = MAKE_COMPILE;
  BY PROGRAM;
RUN;

/* DELETE ORIGINAL MAKE_COMPILE IN DATALIB DIRECTORY */
PROC DATASETS LIB = DATALIB NOWARN NOLIST;
  DELETE MAKE_COMPILE;
RUN;
QUIT;

/* COPY THE UPDATED MAKE_COMPILE TO DATALIB DIRECTORY */
PROC DATASETS LIB = WORK NOLIST;
  COPY OUTLIB = DATALIB;
  SELECT MAKE_COMPILE;
RUN;
QUIT;

/* DELETE EXISTING VERSION IN PRODUCTION */
PROC DATASETS LIB = MACUPD
  MEMTYPE = CATALOG NOWARN NOLIST;
  DELETE SASMACR;
RUN;
QUIT;

/* COPY DEVELOPMENT TO PRODUCTION */
PROC DATASETS LIB = MACDEV
  MEMTYPE = CATALOG NOLIST;
  COPY OUT = MACUPD;
  SELECT SASMACR;
RUN;
QUIT;

%END; /* THERE ARE UPDATES */
%END; /* &ERRCOND = 0 - GOOD RUN */
%MEND DO_COMPILE;

%NOBS MACRO
%macro nobs(data = _LAST_);
  %put NOTE: *** THE MACRO NOBS HAS BEGUN EXECUTION;

  %*****
  /*
  /* MACRO NOBS - FIND NUMBER OF OBSERVATIONS IN A DATASET
  /*
  /*
  /* NOBS IS A MACRO STATEMENT TO DETERMINE THE NUMBER OF
  /* OBSERVATIONS IN A SPECIFIED DATASET.
  /*
  /*

```


Number of Data Set Pages: 1
 Number of Data Set Repairs: 0
 File Name: /home/reidc/blssasug/data/make_compile.sas7bdat
 Release Created: 8.0202M0
 Host Created: SunOS
 Inode Number: 3453185
 Access Permission: rw-rw-r--
 Owner Name: reidc
 File Size (bytes): 24576

The CONTENTS Procedure

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos	Format	Label
4	DTE_UPD	Num	8	0	DATE9.	Date Compiled
3	MAKE	Char	1	288		Compile? Y = Yes/N = No
1	PATH	Char	160	8		Path to Source Program
2	PROGRAM	Char	120	168		Name of Source Program

-----Sort Information-----

Sortedby: PROGRAM
 Validated: YES
 Character Set: ASCII

CONCLUSION

By using an MCD, it is not necessary to add extra lines of code in every macro source file. The programmer can simply turn on or off one or several macro source files by manually changing the value of MAKE to either 'Y' or 'N'. An additional bonus of having an MCD is to allow an authorized user (non-programmers) access to generate an updated compiled macro library without involving a programmer.

REFERENCES

SAS® Online Doc, SAS-L

ACKNOWLEDGMENTS

Mr. Reid wishes to acknowledge his supervisor, Mr. Jürgen Kropf, and coworkers in DIDD in their support of this paper.

CONTACT INFORMATION

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

Curtis E. Reid
 Information Technology Specialist
 Division of Industry Data Development (DIDD)
 Office of Employment and Unemployment Statistics (OEUS)
 Bureau of Labor Statistics (BLS)
 2 Massachusetts Ave., NE Suite 4860
 Washington, DC 20212-0002
reid.curtis@bls.gov

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.