

“Note it and Float It” – How to Use Automatic Macro Variables and E-mail Services for Error Trapping and End User Notification

Bryan K. Beverly, User Technology Associates, Inc., Washington, DC

ABSTRACT

In designing automated SAS applications, it is necessary to capture the errors that do and sometimes do not halt execution, and to notify the appropriate end user that something has gone wrong. While examining the SAS Log is always a reliable option, access to this diagnostic is often restricted to the person who executed the job.

This paper presents some techniques for using automatic macro variables to: (1) capture data step and proc step errors, (2) write the errors out to a text file and (3) use the native e-mail services to notify the appropriate parties, that a specific routine has misfired.

INTRODUCTION

The migration of SAS-based applications from manual to automated execution has introduced a new set of quality control challenges. Jobs whose executions are controlled by host system schedulers can produce anomalies. The user in a timely fashion might not detect the existence of these anomalies. Execution-time error trapping is also a concern for web-based applications, where jobs may run on remote servers and experience undetected data processing, communication or resource problems.

PROBLEM

The problem in managing quality control under these circumstances is that the default error trapping mechanisms function reactively vis-à-vis proactively. The SAS Log is an excellent diagnostic tool, but it only

provides a post-execution description. Moreover, it does not give the user any execution options (i.e., creating execution break points) in the event of errors, nor does it have much flexibility (except when using Proc Printto) in how users are notified that a problem has occurred.

SOLUTION

Using a combination of automatic macro variables (AMVs) and data step techniques, one can manage the execution flow of automated processes. When the SAS system is invoked, the Macro Processor creates AMVs that supply descriptive information about jobs. Some of these AMVs are refreshed after every data step and proc step. Hence, if the values from these AMVs are captured, one can use them to conditionally continue job execution. In like manner, these AMVs can also be used to conditionally create data step routines that write the errors to a text file, and send e-mail messages to the system administrator or end user.

SIGNIFICANCE

The significance of this methodology is that it enables one to create system-driven anomaly management routines that can ‘bullet-proof’ automated applications. Specifically, the user can have total control over how jobs are to execute based upon the values of the AMVs. Hence, instead of waiting for a diagnosis after the fact, one can plan around problems and predetermine if the entire job should proceed to another routine. For example, **&SYSERR** contains the return code set by data steps and procedures. One could branch-off the processing based upon whether the return code was ‘0’ – execution completed successfully and without warning

messages, or '4' – execution completed successfully with warning messages. With respect to tracking and notification, this method not only creates an automated error log but it also facilitates notifying remote key persons that mission critical processing has failed.

CAVEATS

The testing of this approach on both SAS Version 6.12 and SAS Version 8.0 has shown that the data step error trapping under Version 6.12 is not consistent. In those cases, one can create a macro variable, using a CALL SYMPUT with the automatic variable _ERROR_ for the second argument. This technique is handy in general because it enables one to use the automatic variables (AVs) to drive macro variable error trapping. In effect, the pre-execution errors (i.e., syntax errors and semantic errors) as well as execution-time errors and data errors can be assigned to macro variables. Hence, where using the AMVs is not feasible under Version 6.12, the AVs are a viable option. Likewise, combining the two methods under Version 8.0 provides an even wider range of options.

EXAMPLE

To assist in explaining this technique, an example is provided in APPENDIX 1. Assuming a SAS Version 8.0 or higher environment, this routine captures local and remote errors, writes them out to a text file (this is also possible for an HTML file) and sends an e-mail message.

CONCLUSION

This paper has shown how one can manage, document and communicate the execution flow of anomalous automated routines. The reader should be mindful that the approaches and examples presented here are general in scope; each user will have to tailor the code for his or her own processing and reporting needs.

ACKNOWLEDGEMENTS

The author acknowledges the support of User Technology Associates, Inc. and the Division of Current Employment Statistics of the Bureau of Labor Statistics in the development and presentation of this paper.

CONTACT INFORMATION

Bryan K. Beverly
User Technology Associates, Inc.
C/O Bureau of Labor Statistics
2 Massachusetts Avenue, NE
Room 4860
Washington, DC 20212
Voice – 202 691-5466
Fax – 202 691-6644
E-Mail – beverly_b@bls.gov

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries.

APPENDIX 1. – ‘Note It and Float It’ Example

```
/* This code simulates the tasks of downloading a file,
   performing calculations and printing the results.
   Automatic Macro Variables capture the remote and
   local system error code statuses and execute the
   steps based upon the values of the error codes.    */

/* Assign global macro variables that have the values
   from the Automatic Macro Variable created in the
   data step or proc step.    */

%macro alljobs;
  %global check1 check2 check3;

  %macro job1;
    proc download data=mvdata out=pcdata;
    run;
    %sysput check1=&sysinfo; /* capture job status from remote host
  */
%mend job1;                      /* FYI - %sysput is equivalent to %let
*/
%macro job2;
  data one(keep=a b c);
  set pcdata;
  a=1;
  b=2;
  c=a*b/b;
  run;
  %let check2=&syserr; /*capture data step status from local host*/
%mend job2;

%macro job3;
  proc print data=one;
  run;
  %let check3=&syserr; /*capture proc step status from local host*/
%mend job3;

%job1; /* execute the 1st job - if clean then execute the 2nd job
*/
%if &check1=0 %then %do;
  %job2;
%end;

%else %do; /*write error to a job log and send e-mail*/
  filename joblog 'c:\joblog.txt';
  data _null_;
```

```

        file joblog lrecl=80 mod;
        put "JOBID=&sysjobid &sysday &sysdate &systime";
        put "ERROR CODE=&check1";
        put " ";
    run;
    data _null_;
        filename outbox email "somebody@mail.net";
        file outbox
            subject="The Download Blew Up";
        put "JOB &sysjobid blew up on &sysdate at &systime";
        put "Error Code was &check1";
    run;
    endsas; /* end the session if desired */
%end;

/* execute the 2nd job - if clean then execute the 3rd job */
%if &check2=0 %then %do;
    %job3;
%end;

%else %do; /*write error to a job log and send e-mail*/
    filename joblog 'c:\joblog.txt';
    data _null_;
        file joblog lrecl=80 mod;
        put "JOBID=&sysjobid &sysday &sysdate &systime";
        put "ERROR CODE=&check2";
        put " ";
    run;
    data _null_;
        filename outbox email "somebody@mail.net";
        file outbox
            subject="The Calculations Blew Up";
        put "JOB &sysjobid blew up on &sysdate at &systime";
        put "Error Code was &check2";
    run;
    endsas; /* end the session if desired */
%end;

%if &check3^=0 %then %do;
    /*write error to a job log and send e-mail*/
    filename joblog 'c:\joblog.txt';
    data _null_;
        file joblog lrecl=80 mod;
        put "JOBID=&sysjobid &sysday &sysdate &systime";
        put "ERROR CODE=&check3";
        put " ";
    run;
    data _null_;
        filename outbox email "somebody@mail.net";
        file outbox

```

```
        subject="The Print Job Blew Up";
        put "JOB &sysjobid blew up on &sysdate at &systime";
        put "Error Code was &check3";
    run;
    endsas; /* end the session if desired */
%end;

%mend alljobs;
%alljobs;
```


M W S U G

Workshops



