Maximizing and Managing Your SAS® Job Run Capabilities When Using Enterprise Guide® or Standalone PC SAS®

LeRoy Bessler PhD, Bessler Consulting and Research, Strong Smart Systems™
Mequon, Wisconsin, USA, Le_Roy_Bessler@wi.rr.com

Abstract

Enterprise Guide users send code to a remote (or local) server, with no idea of what is happening until the work completes, and the SAS log and any in-client results are displayed. If a remote SAS process is hung, or taking forever for reasons unknown, you probably want to terminate it, but might find the EG option fails. Or, for some reason, you might need to shut down your EG laptop while the remote process is running, which means you will never see the SAS log or your output. Your only option is to kill your EG client session, with unknown consequences for the remote process.

EGbatch lets you run your program in batch mode, from Enterprise Guide or Standalone PC SAS. You get job start, end, and end status emails. They identify your program, and its Windows Process ID (PID). (For EG-submitted code, that PID is NOT the PID for your EG session.)

In batch mode, your log (and any report your code might be creating on disk) are written to your specified disk location, and can be viewed while your job is running. You know what's happening, and how far your job is in its processing.

In a prior version of what I call EGbatch (though it can also be used in PC SAS), the end status email used the return code from the SYSTASK command which launches the bat file that runs your program. During this tool update project, I found that SYSTASK delivers return codes in what can at best be described as an unpredictable fashion. Sometimes a WARNING message triggers Return Code 1, as expected per documentation, but in other cases it delivers a Return Code 0, which is supposed to denote a "clean" run. In other cases, undocumented return codes are delivered.

In any case, the most complete information about how your program ran can be extracted by post-processing to parse its SAS log. This updated and enhanced EGbatch macro has the post-processing log parsing built-in, and sends an End Status Email to report whether or not any anomalies were found, the location of the Anomalies Summary Report (if created), and the location of the full SAS log if you wish to inspect it.

EGbatch not only writes the application program SAS log to disk, but also the SAS log from the parsing of that application program SAS log.

Any report distribution, identifying the location of the report or attaching it, via email, if desired, needs to be built into your application program. That code could make the distribution conditional, but the decision would need to be based on less complete information, not on the results of parsing the SAS log. The report could be distributed unconditionally, as long as the application program verifies that a report (of questionable reliability) exists.

Accessory tools (macros) are provided: (a) **ShowProcessID** to display the PID of your EG session in the SAS log at beginning of the session; (b) **DisplayAllMySASprocesses** to display a list in the SAS log of ALL SAS processes you have running concurrently (if using EG, its PID will be listed also, but not identified as specific to your EG session); and (c) **TerminateProcess** to kill any SAS process that you own (whose PID can first be identified by the display tool). The process display and kill tools are useful even if not using the EGbatch tool. You can kill only processes that you own.

The intended audience for these tools includes not only users of SAS Enterprise Guide (with a remote or a local server), but also users of standalone PC SAS.

Also included is a list of my SAS resource consumption monitoring tools in the bibliography.

Introduction

What Is Delivered In This Paper

The ShowProcessID, DisplayAllMySASprocesses, and TerminateProcess macros are intended for any EG SAS server user or PC SAS user.

The probably misnamed macro EGbatch (which can also be used by PC SAS users) was developed with SAS V9.4 TS1M3 on a 64-bit Windows 7 platform. For SAS Enterprise Guide, Version 7.1 was used, where the full Version information is 7.12 HF2 (7.100.2.3386) (64-bit). The other macros mentioned in the paragraph above were first developed for earlier versions of SAS and EG, but are compatible with the environment used for this project.

In References 1, 2, and 3 are tools for monitoring SAS processes. They can be used by SAS Administrators, SAS managers, and SAS users themselves. Those tools are not essential to the purpose of this paper.

For a tool to help you gain insight into multi-step programs (Is the step spending most time on CPU or on I/O?), see the LogTimer macro in my paper "Supporting Users, Software, and a BI Server: Using SAS to Support SAS", which can be found at: http://www.lexjansen.com/mwsug/2010/resources/MWSUG-2010-163.pdf

Scope and Structure of the Paper

The server operating system is Windows. I don't know whether UNIX analogues to all of these tools can be developed, but I can report that another user developed a UNIX tool to serve the function of my TerminateProcess macro. My tools specifically rely on Windows commands. The only SAS product needed is Base SAS. Macro language is used. ODS can be used to format reports, but it is not used directly by this tool. It can be used in the programs that you run. Macro language and ODS are in Base SAS.

I discuss and demonstrate the three essential accessory macros. Then I discuss and demonstrate the EGbatch macro with eight sample application programs.

NOTE: For each of the test cases, I show the contents of the email inbox, with Process Started message, Process Ended message, and what I call the Process End Status message. They are always generated in that sequence, but Outlook does not necessarily deliver them in that sequence, especially for these simple sample programs which run VERY quickly, and whose SAS logs are parsed VERY quickly. My email software is Microsoft Outlook. My Internet Service Provider is RoadRunner. Not only were the three messages sometimes delivered out of sequence, but also, either my ISP or Outlook sometimes interpreted the third message from the same sending email address in a very short time after two prior from that address as being SPAM, and routed the Process End Status email to my Outlook Junk Mail folder.

I did experiment with launching programs with EGbatch, and terminating them while running:
(a) from a separate EG session while the launching EG session was still open and waiting;
(b) from a separate EG session which, after launch of the program from the first EG session, first terminated that launching EG session and then terminated the launched program; and
(c) from a separate EG session after the launching EG session was terminated manually using Windows Task Manager (which is a proxy for shutting down the launching computer).

This worked fine, but I will not include a description of the scenarios here. I might include it in a future edition of this paper. If interested, send me your email address to request the update if and when becomes available. I should be able to demonstrate these scenarios in the SAS Demo Room at the conference today.

The code for all of the macros is in Appendix 1. The EGbatch macro does not do edits of the invocation parameters. I.e., it assumes that the invoker will do the right thing. Obviously, you can add such improvements.

If you assign to any macro variable in EGbatch a string with an imbedded blank, wrap it with **%STR,**
as in **`MacVar=%STR(Some Value).`**

Tools for What the SAS Enterprise Guide Facilities Do Not Help With

Purpose

The **TerminateProcess** macro is intended for when your process is in one of these situations:

Case 1. Your SAS Enterprise Guide session is no longer available, but the process is still running. This can happen if your PC/laptop has been rebooted or shut down without closing the SAS Enterprise Guide session, or if you used Windows Task Manager to kill SAS Enterprise Guide in desperation because it was frozen.

Case 2. Your SAS Enterprise Guide session is open, but the Stop button and the Task Status Window do not terminate your running process, and you cannot wait for normal termination (or do not think it will ever occur).

SAS provides no way to list all of your concurrent processes on the server. However, to do so, you can use the **DisplayAllMySASprocesses** macro at any time, BUT you cannot run it in an EG session where you already have a submitted process running. You must open a separate EG session.

Run the **ShowProcessID** macro at the start of each EG session and at the start of each PC SAS session. If you have multiple SAS processes running, and decide to terminate one of them, you need to know the identity of each.

How To Use The Above Process Identification / Process Management Tools Their code is in Appendix 1.

These tools were originally designed to be used in the SAS Enterprise Guide Code Window. They can also be used by PC SAS. The ShowProcessID macro can also be used at the beginning of any batch job whose log you are checking during its execution, whether the batch job was launched from Enterprise Guide or via the Windows Task Scheduler or another scheduler such as AutoSys.

1. Using the **ShowProcessID** Macro

In the Code Window, submit this statement:

%ShowProcessID;

Look in the Log Window for feedback. You will get a message of this form:

Process ID for this SAS Enterprise Guide session or SAS batch job is 6212

2. Using the **DisplayAllMySASprocesses** Macro

In the Code Window, submit this statement:

%DisplayAllMySASprocesses;

In the Log Window, look for a listing of your processes. Until this tool is enhanced, the listing will be inelegant, but useful. You can ignore any information of no interest in this context. Key information is the list of all Process IDs for your active SAS processes. (Other information includes the CPU time and Memory used by each process. Scroll to the right in the Log window to see it. CPU time is at the far right.)

3. Using the **TerminateProcess** Macro

(This must be used in an SAS Enterprise Guide session other than the one of the process that you want to terminate.)

In the Code Window, submit this statement:

%TerminateProcess(ProcessID=NNNNNN);

where **NNNNNN** is the Process ID (a.k.a., "PID") for the process ID that you want to terminate.
NNNNNN is typically a three- to six-digit number.

Look in the Log Window for feedback. If your request succeeded, you will get this message:

SUCCESS: Sent termination signal to the process with PID NNNNNN.

Note that it states that a termination signal was **Sent**.

To verify termination, resubmit **%DisplayAllMySASprocesses;**

If you inadvertently try to kill a process that is not yours, the action will fail, and you will get this message (where **ABCDEFGH** will be your user ID):

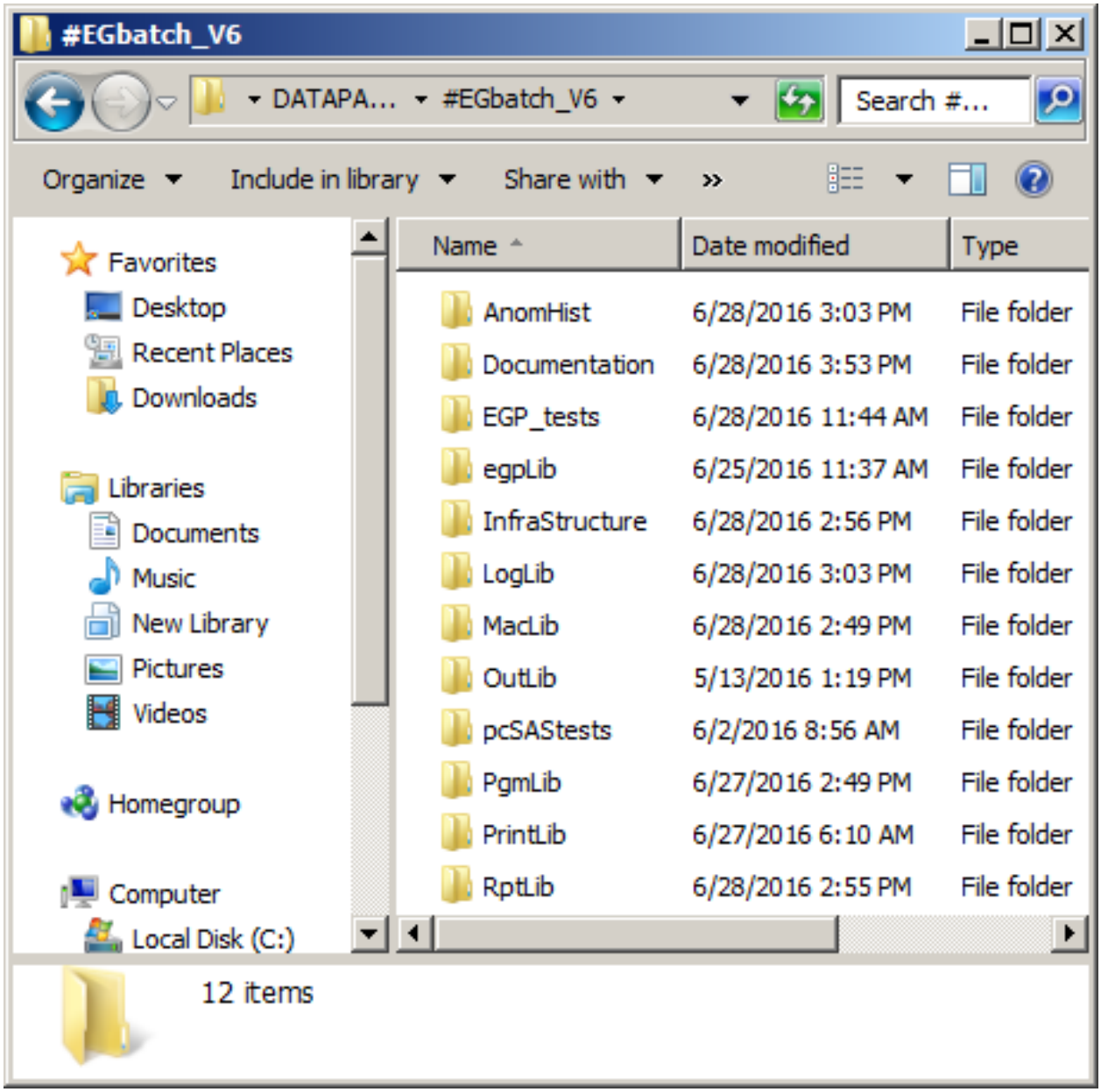
Process ID NNNNNN is not for User ID ABCDEFGH and will not be killed.

If you inadvertently try to kill a non-existent process, the action will fail, and you will get this message:

Process ID NNNNNN was not found.

Folder Structure Supporting Use of the EGbatch Macro

Here is the structure I used during development and testing.



The egplib Folder contains Enterprise Guide projects.

The PgmLib Folder contains the CodeFiles to be run with EGbatch.

The LogLib Folder contains the SAS logs from those EGbatch-launched CodeFile runs.

The RptLib Folder contains reports that are not elsewhere directed by ODS code in the CodeFile.

The EGP_tests Folder contains optionally filed off Code Nodes from the Enterprise Guide projects.

The pcSAS_tests Folder could contain code used to test in PC SAS the EGbatch macro with a CodeFile.

The Documentation Folder contained notes from the development and testing effort.

The OutLib Folder could have optionally contained permanent SAS data sets created by a CodeFile.

The Infrastructure Folder contains: (i) a StoreProcessIDs subfolder that contains the Windows Process IDs that the CodeFiles are run under by the dynamically built Windows bat files; and (ii) artifacts that are **dynamically built** by the EGbatch macro:

(1) a RUN file which is a concatenation of code to:
    a. send a Process Started email
    b. store the Process ID that the RUN file will be run under so it can be retrieved by code outside of the RUN file
    c. capture and pass some context information
    d. run the CodeFile
    e. send a Process Ended email
(2) a JobStart program to send that Process Started email
(3) a GetContext program
(4) a JobEnd program to send that Process Ended email
(5) a BAT file to launch the RUN file

The AnomHist Folder contains:

(a) information about anomalous SAS log records which contain
    Anomaly – short description of the anomaly
    AnomalyCount – frequency of each anomaly, in each record this is equal to 1
    LogLine – SAS log line that contained an anomaly
    LogText – usually or always same as LogLine (probably can be omitted in a future macro rewrite)
    ParseDateTime – when the log parse code was run (always after the CodeFile run end datetime)
    NOTE: Anomaly and AnomalyCount are used to prepare the Summary of Anomalies Report;
(b) Summary of Anomalies Reports; and
(c) SAS logs for every parsing code run

The MacLib Folder contained the evolving versions of the EGbatch macro, as well as the important process identification and termination macros:

(1) ShowProcessID – showing in the SAS log the Windows Process ID for the EG or PC SAS session that submitted the macro
(2) DisplayAllMySASprocesses – showing in the SAS log, of the macro-submitting EG or PC SAS session, ALL of the Windows Process IDs for all SAS executables currently active for the submitting user ID (for a user of DDE to create reports for Excel from SAS, a derivative of this macro could be created to DisplayAllMyExcelProcesses—when using DDE an error situation can leave the Excel server session hung, and needing to be killed)
(3) TerminateProcess – to terminate ANY Windows Process ID belonging to the submitting user ID (there is no verification that the process is for a SAS executable (such an edit could be added)

The PrintLib folder is there as a target for any report output that SAS produces which would be automatically routed to the –PRINT “destination” of the bat file. See Test Case 8 for an example.

There are two alternatives to letting report output go the –PRINT “destination”.

Include an ODS code block in your code file to control format (HTML, PDF, RTF, or LISTING), filename, and physical location of the report output.

Alternatively, the EGbatch macro does provide RptFolder and RptName invocation parameters to control physical location and report filename.

Using the EGbatch Macro to Maximize and Manage Your SAS Job Run Capabilities

Here you will see seven test cases, four that do or try to create reports, and three that do not. Each test case was run from PC SAS and from Enterprise Guide, with identical results. The eighth test case demonstrates using the tool with minimal parameters assigned.

What is presented is:
the application program;
the EGbatch macro invocation to run it:
the email messages that resulted (except ones which are similar in form, with no significant interesting content, to the Start, End, and End Status emails for the first test case): and
the anomaly history summary report when there was any message in the SAS log deemed **by the author** to be important (which, in some cases, is a NOTE, not an ERROR or WARNING). You can, of course, customize the log parsing code to flag additional messages.

For some test cases, less, *but sufficient*, information will be presented to provide adequate understanding.

Test Case 1. PROC PRINT Runs Perfectly

Code:

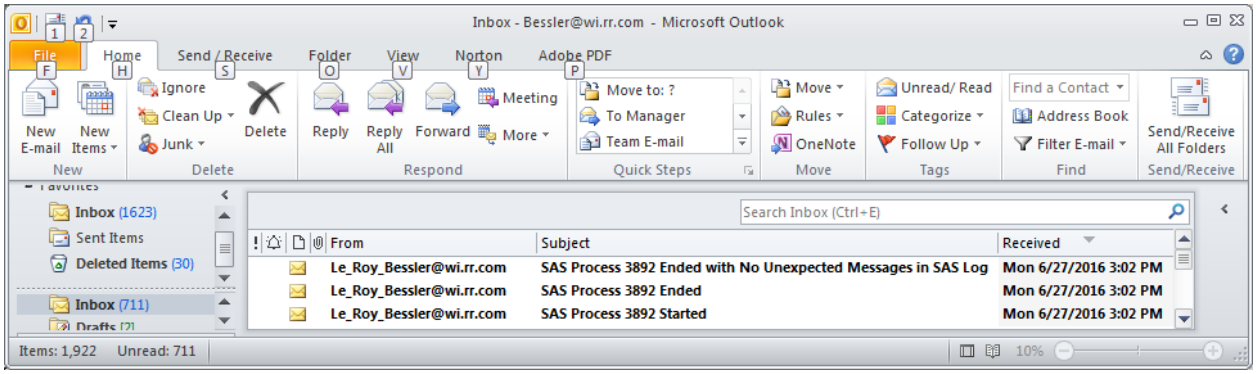
```
options nocenter linesize=max pagesize=max;
ods noresults noproctitle;
ods _all_ close;
ods listing file="&FolderForRpt.\&RptFileName..txt";
            /* If above ODS code is used,
               and running EGbatch with DATETIME=YES, your report
               file name will have a DateTime suffix appended.
               Instead of hard-coding folder and report name here,
               your program picks them up at run time,
               based on EGbatch macro invocation parameters. */
title1 "Listing";
title2 "From Program &FolderForCode.\&CodeFileName";
       /* helpful provenance documentation for the report */
title3 "Run by Process &sysjobid";
title4 "Using OPTIONS nocenter linesize=max pagesize=max";
title5 "This PROC PRINT runs perfectly";
proc print data=sashelp.class; run;
ods listing close; ods listing;
```

Macro Invocation in Enterprise Guide:

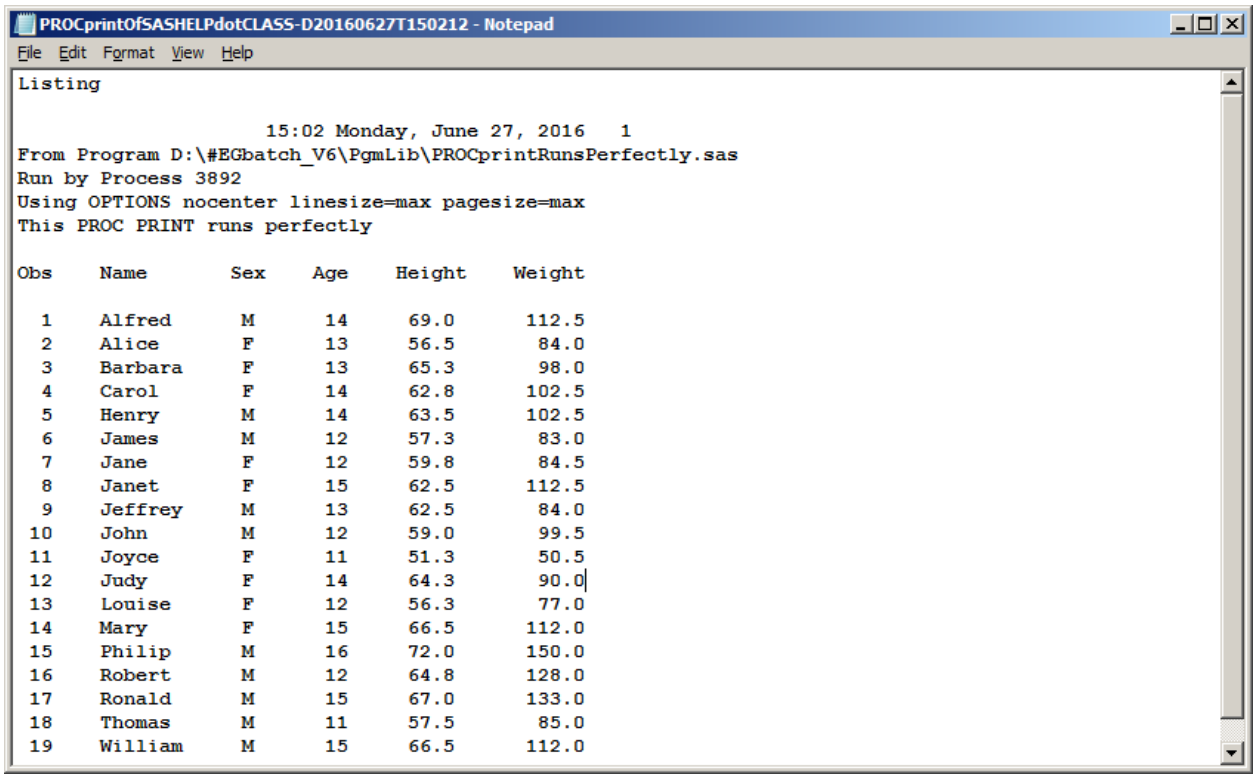
```
options spool; /* not really necessary unless SAS log recommends it,
                  but that means a rerun to get the full SAS log
                  which was incomplete without spool options */
options mprint mprintnest symbolgen mlogic; /* suppress last two? */
options sasautos=("D:\#EGbatch_V6\MacLib" sasautos);
%EGbatch(
exe=%str(C:\Program Files\SASHome\SASFoundation\9.4\sas.exe)
/* above must suit your SAS software path and your SAS version */
,CodeFolder=D:\#EGbatch_V6\PgmLib
,CodeFile=PROCprintRunsPerfectly /* this filename must have extension
                                    .sas and must be in CodeFolder */
```

```
,LogFolder=D:\#EGbatch_V6\LogLib
,DefaultRptFolder=D:\#EGbatch_V6\PrintLib
,RptFolder=D:\#EGbatch_V6\RptLib
,RptName=PROCprintOfSASHELPdotCLASS
,InfraStructureFolder=D:\#EGbatch_V6\InfraStructure
,FolderForAnomalyHistory=D:\#EGbatch_V6\AnomHist
,DateTime=YES
,Notify='Bessler@wi.rr.com'
,FROMemail=Le_Roy_Bessler@wi.rr.com /* This email address can be
                      ANY valid email address that belongs to anyone.
                      Spoofing could get you in trouble, however. */
,Sender=Le_Roy_Bessler@wi.rr.com
,CCemail='cbessie@wi.rr.com'
,BCCemail=
,SupportPerson=%str(LeRoy Bessler)
,SUPPORTemail=Le_Roy_Bessler@wi.rr.com);
```

Email InBox of Notify Recipient:

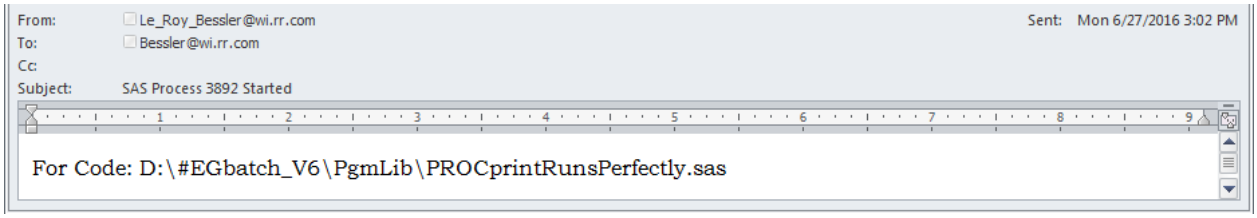


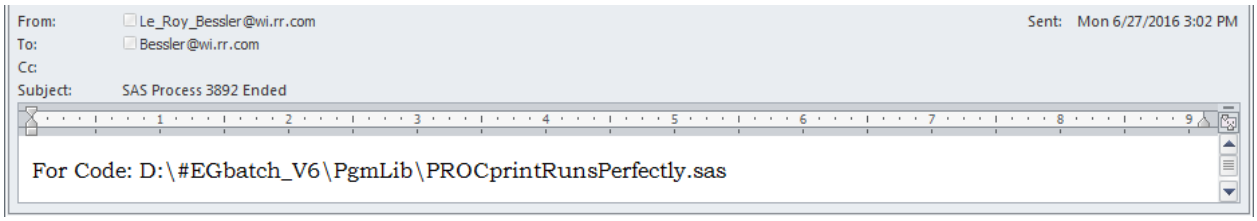
Report Result:

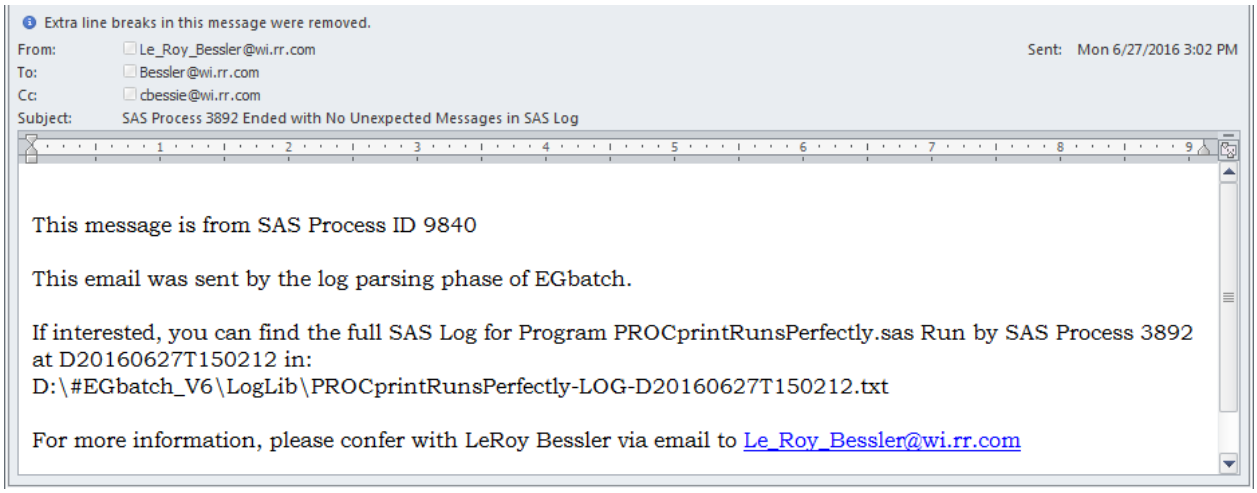


Because there were no unexpected messages, there is no Summary of Anomalies Report.

Here are the contents of the three emails:







Test Case 2. PROC PRINT Runs With A Bad LABEL Statement

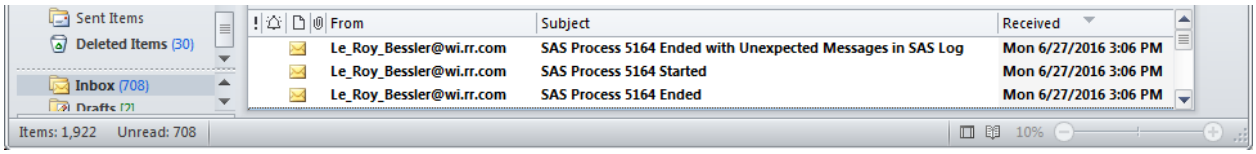
Code:

```
options nocenter linesize=max pagesize=max;
ods noresults;
ods noproctitle;
ods _all_ close;
ods listing file="&FolderForRpt.\&RptFileName..txt";
title1 "Listing";
title2 "From Program &FolderForCode.\&CodeFileName";
title3 "Run by Process &sysjobid";
title4 "Using OPTIONS nocenter linesize=max pagesize=max";
title5 "This PROC PRINT will run, but with a WARNING message";
proc print data=sashelp.class;
label Gender='Sex'; /* Gender is not present in SASHELP.CLASS */
run;
ods listing close;
ods listing;
```

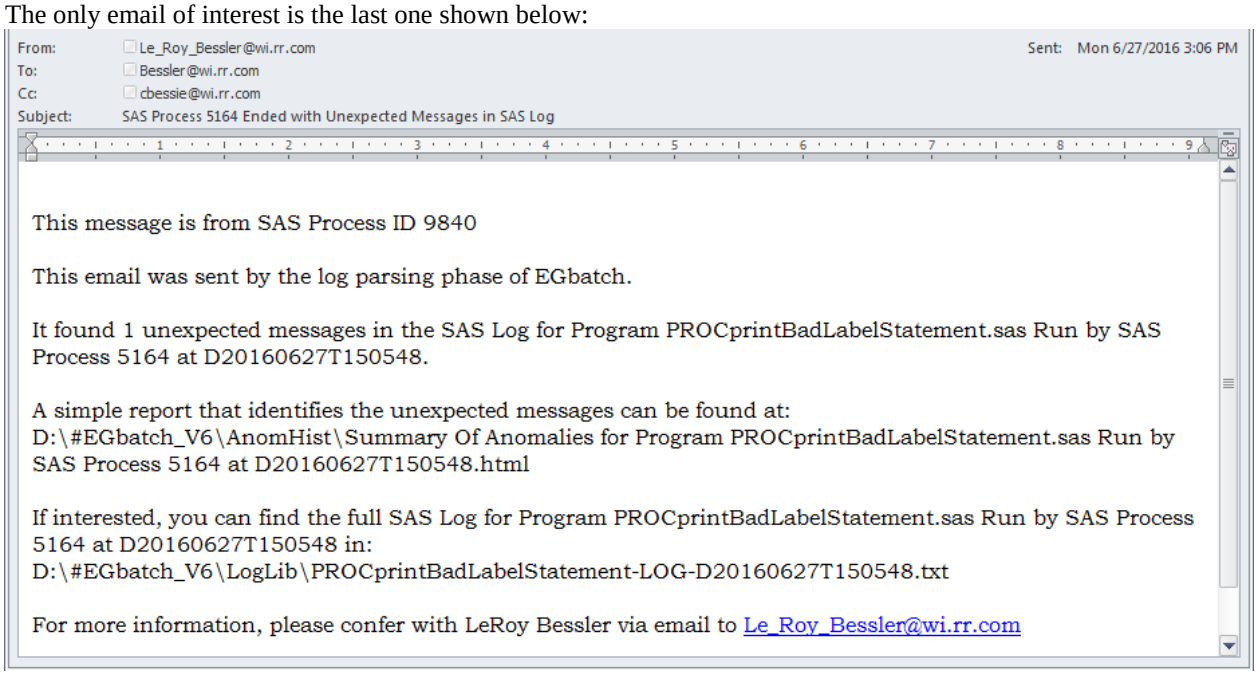
Macro Invocation in Enterprise Guide:

```
options spool;
options mprint mprintnest symbolgen mlogic;
options sasautos=("D:\#EGbatch_V6\MacLib" sasautos);
%EGbatch(
/* NOTE: exe= is my site macro default. Not explicitly coded here. */
CodeFolder=D:\#EGbatch_V6\PgmLib
,CodeFile=PROCprintBadLabelStatement
,LogFolder=D:\#EGbatch_V6\LogLib
,DefaultRptFolder=D:\#EGbatch_V6\PrintLib
,RptFolder=D:\#EGbatch_V6\RptLib
,RptName=PROCprintOfSASHELPdotCLASS
,InfraStructureFolder=D:\#EGbatch_V6\InfraStructure
,FolderForAnomalyHistory=D:\#EGbatch_V6\AnomHist
,DateTime=YES
,Notify='Bessler@wi.rr.com'
,FROMemail=Le_Roy_Bessler@wi.rr.com
,Sender=Le_Roy_Bessler@wi.rr.com
,CCemail='cbessie@wi.rr.com'
,BCCemail=
,SupportPerson=LeRoy Bessler
,SUPPORTemail=Le_Roy_Bessler@wi.rr.com);
```

Email InBox (first and second messages appear out of sequence):



The only email of interest is the last one shown below:



From: Le_Roy_Bessler@wi.rr.com
To: Bessler@wi.rr.com
Cc: cbessie@wi.rr.com
Subject: SAS Process 5164 Ended with Unexpected Messages in SAS Log
Sent: Mon 6/27/2016 3:06 PM

This message is from SAS Process ID 9840

This email was sent by the log parsing phase of EGbatch.

It found 1 unexpected messages in the SAS Log for Program PROCprintBadLabelStatement.sas Run by SAS Process 5164 at D20160627T150548.

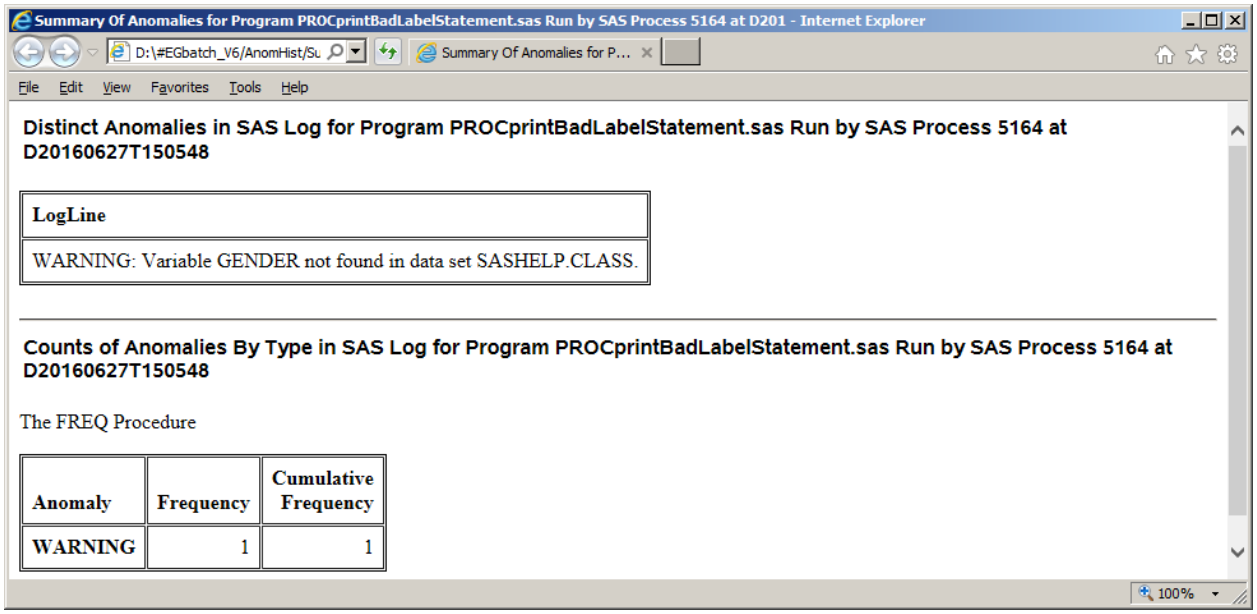
A simple report that identifies the unexpected messages can be found at:
D:\#EGbatch_V6\AnomHist\Summary Of Anomalies for Program PROCprintBadLabelStatement.sas Run by SAS Process 5164 at D20160627T150548.html

If interested, you can find the full SAS Log for Program PROCprintBadLabelStatement.sas Run by SAS Process 5164 at D20160627T150548 in:
D:\#EGbatch_V6\LogLib\PROCprintBadLabelStatement-LOG-D20160627T150548.txt

For more information, please confer with LeRoy Bessler via email to Le_Roy_Bessler@wi.rr.com

The Report Result matches that of Test Case 1, since the Label statement is for a non-present variable.

Here is the Anomalies Summary Report referrred to in the Process End Status email above:



Summary Of Anomalies for Program PROCprintBadLabelStatement.sas Run by SAS Process 5164 at D201 - Internet Explorer

File Edit View Favorites Tools Help

Distinct Anomalies in SAS Log for Program PROCprintBadLabelStatement.sas Run by SAS Process 5164 at D20160627T150548

LogLine
WARNING: Variable GENDER not found in data set SASHELP.CLASS.

Counts of Anomalies By Type in SAS Log for Program PROCprintBadLabelStatement.sas Run by SAS Process 5164 at D20160627T150548

The FREQ Procedure

Anomaly	Frequency	Cumulative Frequency
WARNING	1	1

Test Case 3. PROC PRINT Fails With A Bad FORMAT Statement

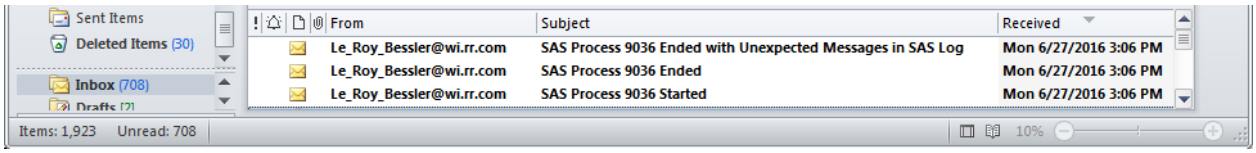
Code:

```
options nocenter linesize=max pagesize=max;
ods noresults;
ods noproctitle;
ods _all_ close;
ods listing file="&FolderForRpt.\&RptFileName..txt";
title1 "Listing";
title2 "From Program &FolderForCode.\&CodeFileName";
title3 "Run by Process &sysjobid";
title4 "Using OPTIONS nocenter linesize=max pagesize=max";
title5 "This PROC PRINT will run, but with a WARNING message";
proc print data=sashelp.class;
format Sex 1.; /* The Sex variable is character, M or F */
run;
ods listing close;
ods listing;
```

Macro Invocation in Enterprise Guide:

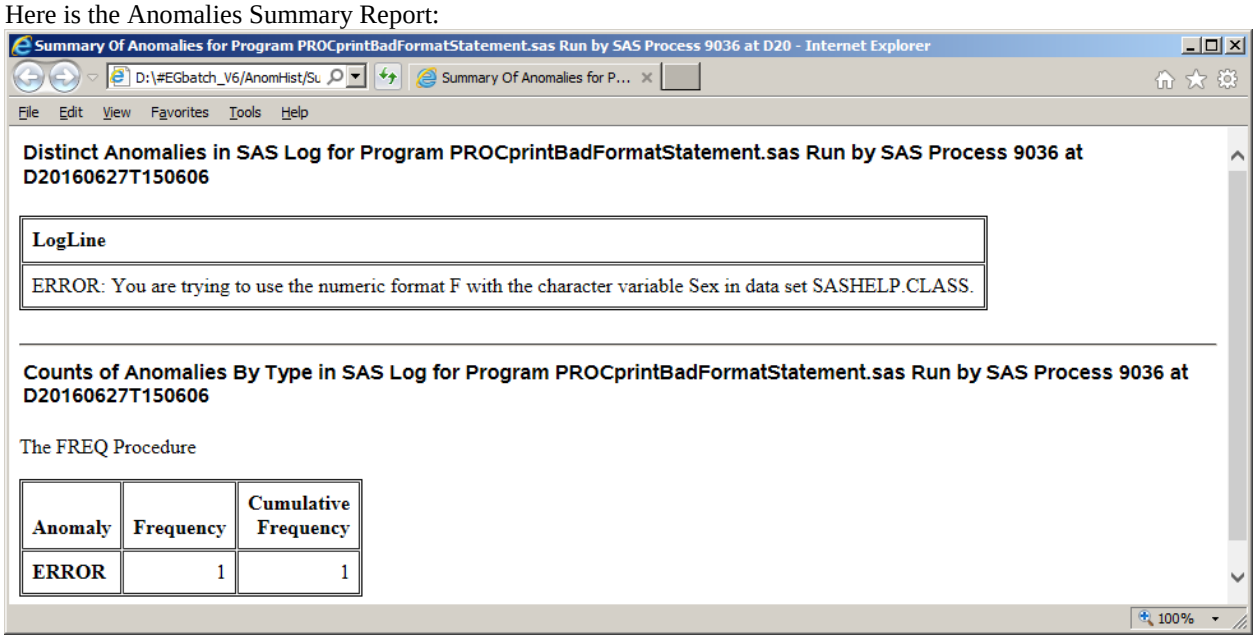
```
options spool;
options mprint mprintnest symbolgen mlogic;
options sasautos=("D:\#EGbatch_V6\MacLib" sasautos);
%EGbatch(CodeFolder=D:\#EGbatch_V6\PgmLib
,CodeFile=PROCprintBadFormatStatement
,LogFolder=D:\#EGbatch_V6\LogLib
,DefaultRptFolder=D:\#EGbatch_V6\PrintLib
,RptFolder=D:\#EGbatch_V6\RptLib
,RptName=PROCprintOfSASHELPdotCLASS
,InfraStructureFolder=D:\#EGbatch_V6\InfraStructure
,FolderForAnomalyHistory=D:\#EGbatch_V6\AnomHist
,DateTime=YES
,Notify='Bessler@wi.rr.com'
,FROMemail=Le_Roy_Bessler@wi.rr.com
,Sender=Le_Roy_Bessler@wi.rr.com
,CCemail='cbessie@wi.rr.com'
,BCCemail=
,SupportPerson=LeRoy Bessler
,SUPPORTemail=Le_Roy_Bessler@wi.rr.com);
```

Email InBox :



The Report Result is an empty file, since PROC PRINT ended with an ERROR as reported in the Anomalies Summary Report referred to in the Process End Status email.

Here is the Anomalies Summary Report:



Test Case 4. PROC PRINT Fails With A Bad VAR Statement

Code:

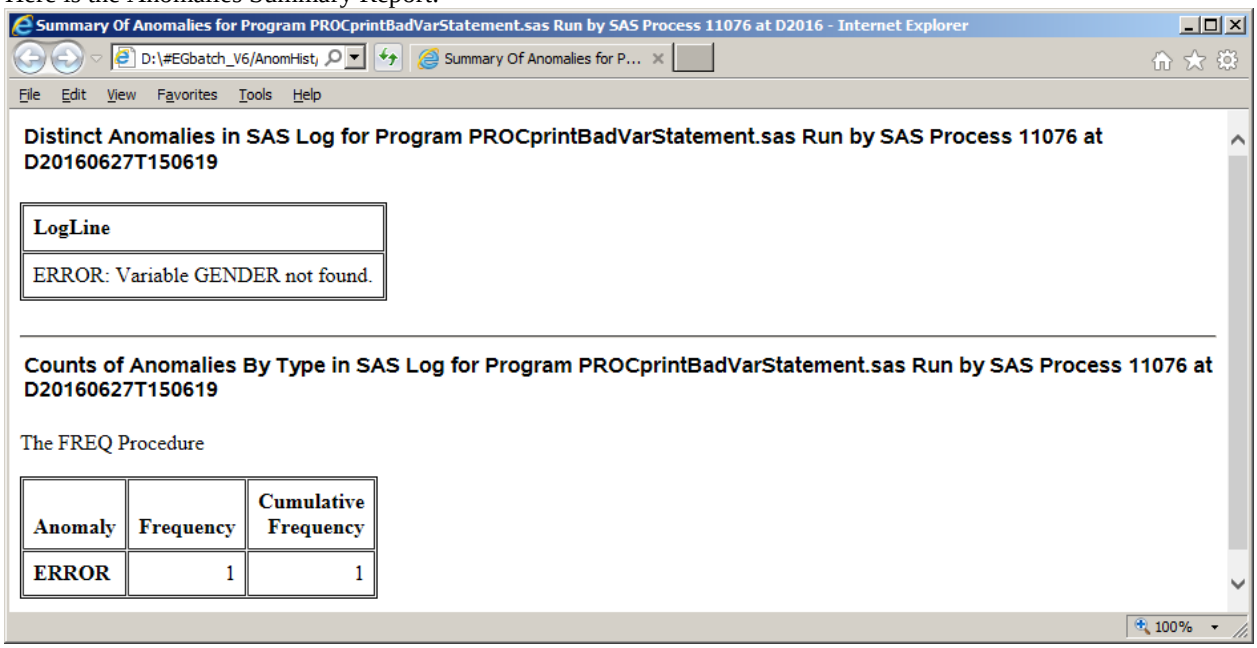
```
options nocenter linesize=max pagesize=max;
ods noresults;
ods noproctitle;
ods _all_ close;
ods listing file="&FolderForRpt.\&RptFileName..txt";
title1 "Listing";
title2 "From Program &FolderForCode.\&CodeFileName";
title3 "Run by Process &sysjobid";
title4 "Using OPTIONS nocenter linesize=max pagesize=max";
title5 "This PROC REPORT will fail, triggering an ERROR message";
proc print data=sashelp.class;
var Name Gender Age Height Weight;
      /* Sex, not Gender, is in SASHELP.CLASS */
run;
ods listing close;
ods listing;
```

Macro Invocation in Enterprise Guide:

Same as for Test Cases 1, 2, and 3, except **`CodeFile=PROCprintBadVarStatement`**

Report Result is an empty file, since PROC PRINT ended with an ERROR.

Here is the Anomalies Summary Report:



The Process End Status email is similar to that of Test Case 3.

Test Case 5. Probably A Bad Merge

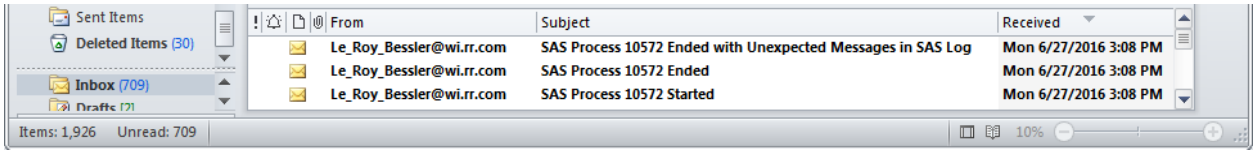
Code (prepares fake two fake data sets, each with non-unique keys):

```
options nocenter linesize=max pagesize=max;
data work.MaleStudents_AgeHgt_1(drop=Sex);
set sashelp.class(keep=Name Sex Age Height);
if _N_ EQ 19 then Name = 'Alfred';
rename Age = Age1;
rename Height = Height1;
if Sex EQ 'F' then delete; run;
data work.MaleStudents_AgeHgt_2(drop=Sex);
set sashelp.class(keep=Name Sex Age Height);
if _N_ EQ 15 then Name = 'Alfred';
rename Age = Age2;
rename Height = Height2;
if Sex EQ 'F' then delete; run;
proc sort data=work.MaleStudents_AgeHgt_1; by Name; run;
proc sort data=work.MaleStudents_AgeHgt_2; by Name; run;
data work.MaleStudents_HgtWithAging;
merge work.MaleStudents_AgeHgt_1(in=EarlierAge)
work.MaleStudents_AgeHgt_2(in=LaterAge);
by Name;
if EarlierAge and LaterAge; run;
```

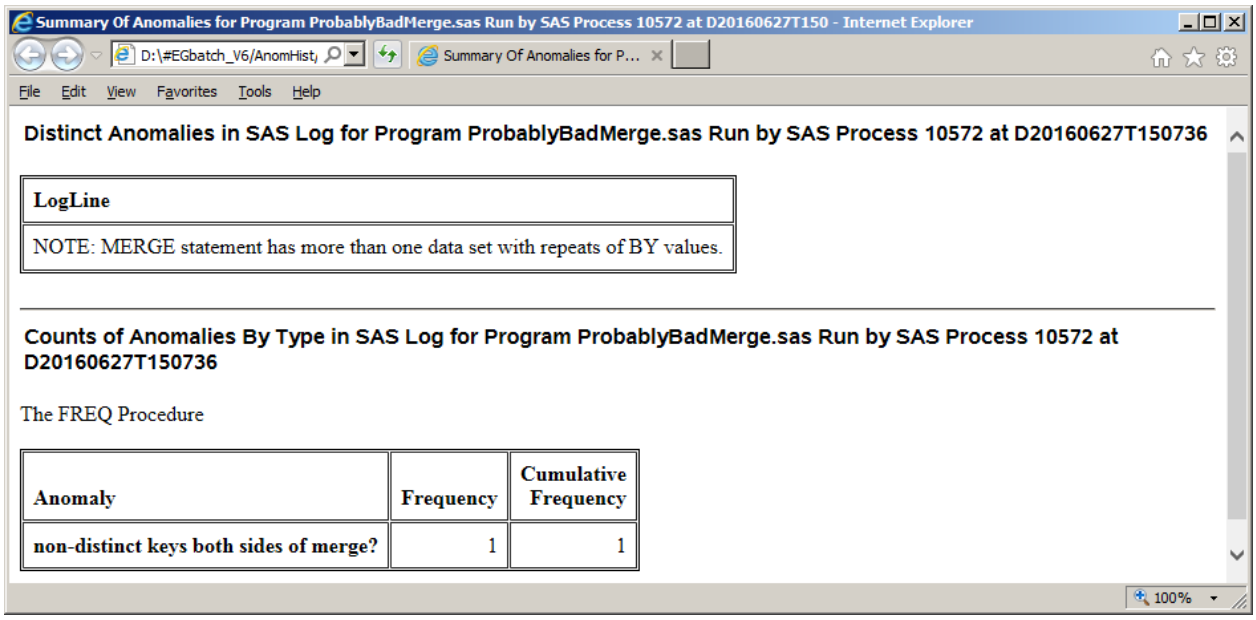
Macro Invocation in Enterprise Guide:

```
options spool;
options mprint mprintnest symbolgen mlogic;
options sasautos=("D:\#EGbatch_V6\MacLib" sasautos);
%EGbatch(CodeFolder=D:\#EGbatch_V6\PgmLib
,CodeFile=ProbablyBadMerge
,LogFolder=D:\#EGbatch_V6\LogLib
,DefaultRptFolder=D:\#EGbatch_V6\PrintLib
,InfraStructureFolder=D:\#EGbatch_V6\InfraStructure
,FolderForAnomalyHistory=D:\#EGbatch_V6\AnomHist
,DateTime=YES
,Notify='Bessler@wi.rr.com'
,FROMemail=Le_Roy_Bessler@wi.rr.com
,Sender=Le_Roy_Bessler@wi.rr.com
,CCemail='cbessie@wi.rr.com'
,BCCemail=
,SupportPerson=LeRoy Bessler
,SUPPORTemail=Le_Roy_Bessler@wi.rr.com);
```

Email InBox (all three messages appear in sequence):



Here is the Anomalies Summary Report:



Why a two-part report, rather than just a listing of ALL anomalies? Well, unlike the examples shown in this paper, anomalies of a particular type or types might occur multiple times. It's not worthwhile to list all of the "duplicates", but the summary is a guide for the process of actually browsing the SAS log to understand what's happening.

Test Case 6. DATA Step with variable uninitialized (not uncommon before coding is complete and correct)

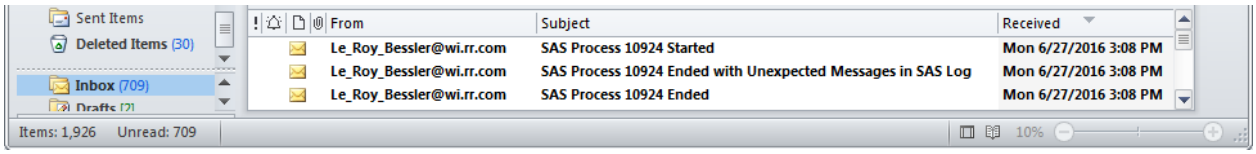
Code:

```
options nocenter linesize=max pagesize=max;
data work.ClassButExtraVarUninitialized;
set sashelp.class;
label City="Student's City";
run;
```

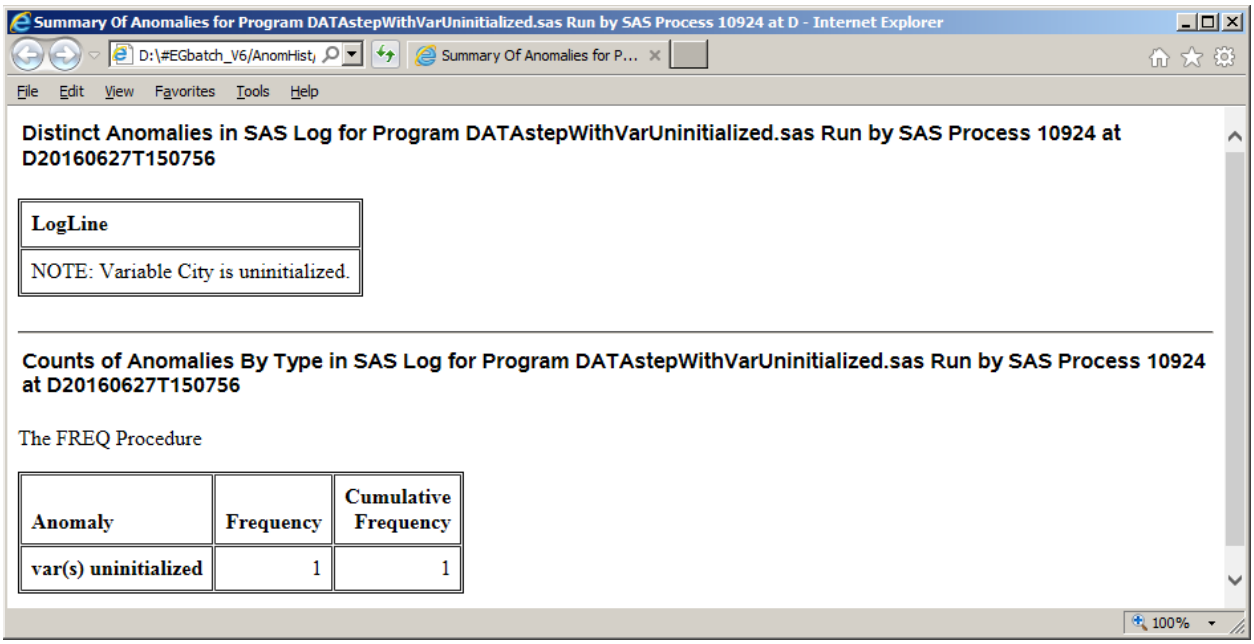
Macro Invocation in Enterprise Guide:

Same as for Test Case 5, except **`CodeFile=DATAstepWithVarUninitialized`**

Email InBox (first message arrived last):



Here is the Anomalies Summary Report:



Sometimes code, usually through oversight, contains a leftover reference to a never used variable, as in the case of this fake example, or a no longer used variable Even if the leftover reference has no harmful effect on execution of the program, it is always best practice to have finished code with no stray irrelevant leftovers creating potentially confusing messages in the log.

Test Case 7. DATA Step with variable not referenced (not uncommon before coding is complete and correct)

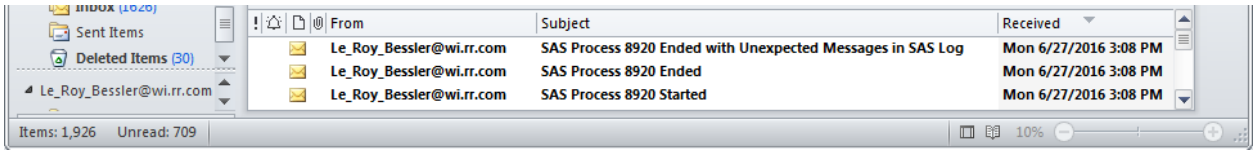
Code:

```
options nocenter linesize=max pagesize=max;
data work.ClassButNonVarCannotBeKept
(keep=Name Sex Age Height Weight School);
set sashelp.class;
run;
```

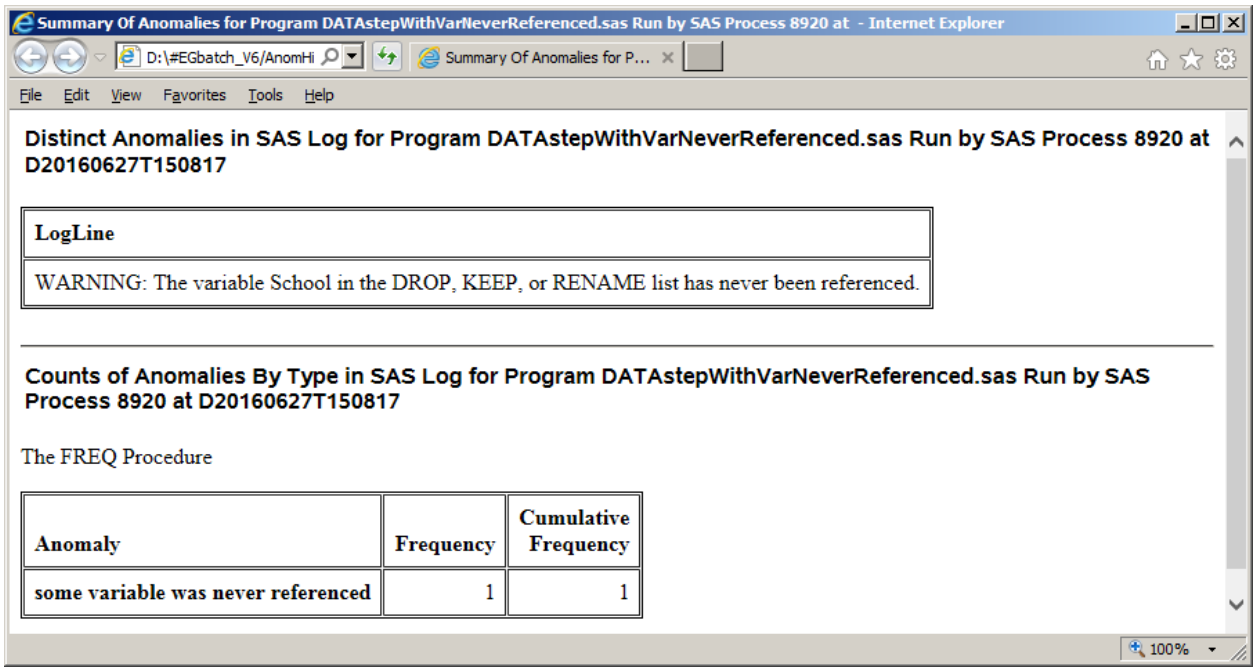
Macro Invocation in Enterprise Guide:

Same as for Test Case 5, except `,CodeFile=DATAstepWithVarNeverReferenced`

Email InBox (first and second message appear in reverse order):



Here is the Anomalies Summary Report:



Test Case 8. Using the tool with minimal parameters assigned.

Code:

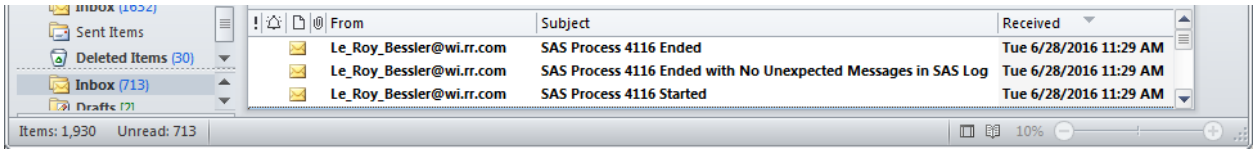
```
data work.ToPROCprint;
length line $ 6;
line = 'Hello,'; output;
line = 'SAS';    output;
line = 'Users!'; output;
run;
title;
options nocenter nodate nonumber;
proc print data=work.ToPROCprint noobs;
label line='Greeting';
run;
```

Macro Invocation in Enterprise Guide:

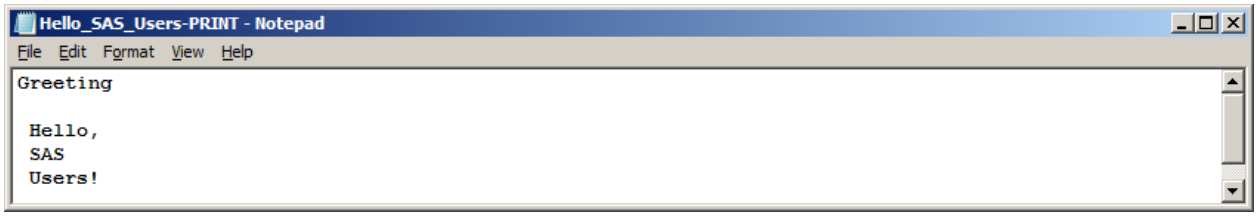
```
options spool; /* prevents possibly incomplete SAS log */
options mprint mprintnest; /* recommended */
options symbolgen mlogic;  /* optional */
options sasautos=("D:\#EGbatch\MacLib" sasautos);

%EGbatch(
CodeFolder=D:\#EGbatch_V6\PgmLib
,CodeFile=Hello_SAS_Users /* this filename must have extension
                             .sas and must be in CodeFolder */
,LogFolder=D:\#EGbatch_V6\LogLib
,DefaultRptFolder=D:\#EGbatch_V6\PrintLib /* important for Case 8 */
,InfraStructureFolder=D:\#EGbatch_V6\InfraStructure
,FolderForAnomalyHistory=D:\#EGbatch_V6\AnomHist
,Notify='Bessler@wi.rr.com'
,FROMemail=Le_Roy_Bessler@wi.rr.com /* This email address can be
                      ANY valid email address that belongs to anyone.
                      Spoofing could get you in trouble, however. */
);
```

Email InBox (second and third messages appear in reverse order):



There is NO Anomalies Summary Report since there were No Unexpected Messages. Here is the report output that is routed to the DefaultRptFolder:



Using SAS to Monitor SAS Users and Processes

Rather than extend this already long paper with excerpts from prior work, I simply suggest that you investigate References 1, 2, and 3. These tools filled a void not addressed by tools from SAS Institute.

Conclusion

EGbatch, which has also has benefits for users of PC SAS, gets the processing of your code submitted from Enterprise Guide out of the dark. You can browse the SAS log on disk while your program is running, rather than being mystified about why it's running so long and wondering what it is happening. And the job run helpfully sends you Start, End, and End Status emails, with program name (in the email body) and the SAS Process ID being identified. If, due to what you learn by inspection of the SAS log on disk for a running job, or for any other reason, you wish to prematurely end your job, the TerminateProcess macro gives you that power. To be able to identify which SAS process to terminate (you will have the EG session active that launched the codefile with EGbatch, and an EG session from which to submit Terminate Process. You use ShowProcessID in each at start of EG session to self-identify, and DisplayAllMySASprocesses to identify the process ID for your running CodeFile.

With its automatic parsing of the SAS log, use of EGbatch can accelerate development by directly disclosing where any problems might be in the code. Any suggestions for improving the parsing are welcome.

References

1. Bessler, LeRoy. "Using SAS to Manage, Monitor, and Control the SAS BI Server: User-Developed Custom Tools for the SAS Server Administrator, User, or Manager", *Proceedings of the SAS Global Forum 2009 Conference*. Find it on the web at http://support.sas.com/resources/papers/proceedings09/274-2009.pdf
2. Bessler, LeRoy. "More Ways to Use SAS to Manage, Monitor, and Control SAS or the SAS BI Server: Tools for the SAS User, Server Administrator, or Manager", *Proceedings of the SAS Global Forum 2010 Conference*. Find it on the web at http://support.sas.com/resources/papers/proceedings10/279-2010.pdf
3. Bessler, LeRoy and Andruskevitch, Victor. "Using SAS to Manage, Monitor, and Control the SAS BI Server: User-Developed Custom Tools for the SAS Server Administrator, User, or Manager", *Proceedings of the SAS Global Forum 2012 Conference*. Find it on the web at http://support.sas.com/resources/papers/proceedings12/356-2012.pdf

Contact Information, Etc.

Your comments, questions, and suggestions for improvements or enhancements are welcome.

LeRoy Bessler PhD
Le_Roy_Bessler@wi.rr.com
Bessler Consulting and Research
Strong Smart Systems™
Visual Data Insights™
Mequon, Wisconsin, USA

Ancora Imparo
(I am still learning)
— Michelangelo, on his 87th birthday

A SAS user since 1978, Dr. LeRoy Bessler has shared his knowledge and experience with other users at conferences throughout the USA and in Montreal, London, Heidelberg, and Dublin. Though a SAS generalist with long experience in Base SAS, SAS macro language, and SAS tools for access to non-SAS data, his special interests include creation of unique tools to support the SAS BI server and its users, communication-effective visual communication and reporting, web information delivery, highly formatted Excel reporting, SAS/GRAPH, ODS, and Software-Intelligent Application Development for Reliability, Reusability, Extendibility, Maintainability, and Flexibility. If interested, send him an email request for an index to all of his SAS papers, presentations, and VIEWS Newsletter articles that are available via the internet.

Appendix 1.

EGbatch Macro

```
/* Run this macro from PC SAS or from Enterprise Guide. At the start of either
session, run the ShowProcessID macro to be able to distinguish its SAS process from
that the program that you will launch. To see ALL SAS processes that you might have
running at any time, run the DisplayAllMySASprocesses macro. */

/* This macro has been tested, but not every possible combination of macro parameter
assignments. Also, the values assigned to macro parameters are NOT edited. The macro
assumes that you will use it correctly. If you discover any coding errors, or can
provide any coding improvements, kindly share that information with the author. */

/* If you run the macro with DATETIME=YES (the default), you will accumulate datetime-
stamped files in LogFolder, InfrastructureFolder, FolderForAnomalyHistory, RptFolder,
and possibly DefaultRptFolder. Suppressing accumulation of certain infrastructure
files can be accomplished by uncommenting a pair of delete statements in the macro. In
the unlikely event that you are running the same program in two or more sessions
concurrently, you should use DateTime=YES. */

/* If you have questions about this macro, or if you wish to receive future updates,
please notify the author via email. */

%macro EGbatch (
exe=%str(C:\Program Files\SASHome\SASFoundation\9.4\sas.exe) /* adjust
this to suit the machine where SAS is running and your SAS version */
,CodeFolder= /* mandatory */
,CodeFile=   /* mandatory */
,LogFolder=  /* mandatory */
,DefaultRptFolder= /* mandatory
                      If a PROC creates report output,
                      and you have not packaged that output
                      with ODS in the CodeFile,
                      a txt file is written to this folder. */
,RptFolder=  /* optional, useful to specify
                an application-specific report folder */
,RptName=    /* if omitted, default is CodeFile */
,InfraStructureFolder=    /* mandatory */
,FolderForAnomalyHistory= /* mandatory
                             also contains SAS logs from parsing
                             the CodeFile run SAS logs */
,DateTime=NO    /* If YES, things will acccumulate,
                   and you will need to do clean-up from time to time.
                   With NO, you will not be able to save old runs,
                   and distinguish them from one another.
                   What could have been datetimestamped
                   will get overwritten with each run. */
,Notify=        /* mandatory */
,CCemail=       /* optional */
,BCCemail=      /* optional */
 /* SAS verifies the validity of any outbound email addresses used */
,FROMemail=     /* mandatory
                   This email address can be
                   ANY valid email address that belongs to anyone.
```

```
                  Spoofing could get you in trouble, however. */
,SupportPerson= /* optional */
,SUPPORTemail=  /* optional,
                  but used only if SuppportPerson is non-null */
);

 /* In the list of invocation macro variables above,
    assign as many defaults as possible
    for all of those that will always be the same.
    Then your invocation of the macro
    only needs to specify the ones that vary. */

*************************************************************************;
* Macro:   EGbatch.sas *;
* Created: 28 June 2016 *;
* Author:  LeRoy Bessler PhD *;
* Firm:    Bessler Consulting and Research *;
* Email:   Le_Roy_Bessler@wi.rr.com *;
* Comments: *;
* Updated On: *;
* Updated By: *:
* Reason: *;
*************************************************************************;

%macro PassContext;

data _null_;
length line $ 256;
file "&InfrastructureFolder.\GetContext.sas" lrecl=256;
line = '%let CodeFileName  = ' || "&CodeFile..sas" || ";";
put line $256.;
line = '%let FolderForCode = ' || "&CodeFolder" || ";";
put line $256.;
line = '%let FolderForRpt  = ' || "&RptFolder" ||";";
put line $256.;
line = '%let RptFileName   = '
%if %length(&RptName) NE 0 %then %do;
                               || "&RptName"
%end;
%else %do;
                               || "&CodeFile"
%end;
%if &DateTime EQ YES %then %do;
                               || "-&DT"
%end;
                               ||";";
put line $256.;
run;

%mend  PassContext;

%macro JobMsg(StartOrEnd=);
```

```
data _null_;
length line $ 256;
file "&InfrastructureFolder.\Job&StartOrEnd..sas" lrecl=256;
line = 'FILENAME SASMAIL EMAIL' ||
'  TO="'     || &Notify      || /* allow multiple email addresses */
'" FROM="'   || "&FROMemail" ||
'" SENDER="' || "&FROMemail" ||
'" SUBJECT="SAS Process ' || '&sysjobid' ||
" &StartOrEnd.ed" || '";';
put line $256.;
line = 'data _null_;';
put line $256.;
line = 'file SASMAIL;';
put line $256.;
line = "put 'For Code: " || "&Code.';";
put line $256.;
line = 'run;';
put line $256.;
run;

%mend  JobMsg;

%macro StorePIDforStatusEmail;

data _null_;
length line $ 256 /* PID $ 16 */ InfrastructureFolder $ 128;
file "&InfrastructureFolder.\StorePIDforStatusEmail.sas" lrecl=256;
InfrastructureFolder = "&InfrastructureFolder";
line = 'libname StorePID "' || trim(left(InfrastructureFolder)) ||
'\StoreProcessIDs";';
put line $256.;
line = "data StorePID.PID_&DT;";
put line $256.;
line = 'PID = "' || '&sysjobid' || '";';
put line $256.;
line = 'run;';
put line $256.;
run;

%mend  StorePIDforStatusEmail;

%global Program;
%let Program = &CodeFile..sas;
%put Program is &Program;

%global DT;

libname StorePID "&InfrastructureFolder.\StoreProcessIDs";

data _null_;
length DateTime $ 16 Time $ 6;
```

```
Time = trim(left(compress(put(time(),time8.),':')));
if length(Time) EQ 5 then Time = '0' || left(Time);
DateTime = 'D' ||
trim(left(put(today(),yymmddn8.))) ||
 'T' || Time;
call symput('DT',DateTime);
run;

%let Code = &CodeFolder.\&CodeFile..sas;

%let DateTime = %upcase(&DateTime);

%let Work = %substr(%sysfunc(pathname(work)),1,
  %eval(%index(%sysfunc(pathname(work)),_) - 2));

%if &DateTime EQ YES
%then %let FileSuffix = -&DT;
%else %let FileSuffix = %str();
%let Bat = &InfrastructureFolder.\&CodeFile.-BAT&FileSuffix..bat;
%let Run = &InfrastructureFolder.\&CodeFile.-RUN&FileSuffix..sas;
%let ApplPgmLog = &LogFolder.\&CodeFile.-LOG&FileSuffix..txt;
%let DefaultRpt =
           &DefaultRptFolder.\&CodeFile.-PRINT&FileSuffix..txt;

%global SASlogToBeParsed;
%let SASlogToBeParsed = &ApplPgmLog;
%put SASlogToBeParsed is &SASlogToBeParsed;

%JobMsg(StartOrEnd=Start);
%StorePIDforStatusEmail;
%JobMsg(StartOrEnd=End);
%PassContext;

filename concat2
(
"&InfrastructureFolder.\JobStart.sas"
"&InfrastructureFolder.\StorePIDforStatusEmail.sas"
"&InfrastructureFolder.\GetContext.sas"
"&Code"
"&InfrastructureFolder.\JobEnd.sas"
);

data _null_;
infile concat2; input;
file "&Run"; put _infile_;
run;

options noxwait noxsync;

* x "del &InfrastructureFolder.\JobStart.sas";
* x "del &InfrastructureFolder.\JobEnd.sas";
  /* Not important to delete. They get overwritten with every run. */
```

```
data _null_;
length line $ 1024;
file "&Bat" lrecl=1024;
line =
'"' || "&exe" || '"'  ||
' -sysin "' || "&Run"  || '"' ||
' -work "'  || "&Work" || '"' ||
' -log "'   || "&ApplPgmLog"  || '"' ||
' -print "' || "&DefaultRpt"  || '"' ||
' -unbuflog -nosplash -nologo -icon';
 /* unbuflog forces writes to the SAS log more often,
    rather than permitting SAS to buffer (hold) log content
    and send chunks out later. However, when SAS starts
    a DATA step or PROC step, final messages pertaining to it
    do not get written to the log until end of step. */
put line $1024.;
line = 'EXIT';
put line $1024.;
run;

systask command "&Bat" wait status=TaskRC;
     /* &TaskRC no longer being used later due to erratic results
        for SYSTASK return codes. */

data _null_;
length PID $ 16;
set StorePID.PID_&DT;
call symput('LaunchedProcessID',trim(left(PID)));
run;

%put &LaunchedProcessID is &LaunchedProcessID;

data _null_; x=sleep(1); run; /* present here in case
                                 subsequent deletes are UnCommented */

* x "del &Run"; /* UnComment this to avoid accumulation
                   of Code Run SAS files if using DateTime=YES */
* x "del &Bat"; /* UnComment this to avoid accumulation
                   of BAT files if using DateTime=YES */

options linesize=max pagesize=max;

data _null_;
length ParseDateTime $ 14 TimePart $ 6;
TimePart = left(compress(put(time(),time8.),':'));
if length(TimePart) EQ 5
then TimePart = '0' || trim(left(TimePart));
ParseDateTime = put(today(),date7.) || '_' || TimePart;
call symput('ParseDateTime',ParseDateTime);
run;
```

```
proc printto
log="&FolderForAnomalyHistory.\SASlogForParseOfSASlogForProgram_&Progr
am._Run_At_&DT..txt";
run;

options nosource;
%put ******************************************************;
%put ******************************************************;
%put ******************************************************;
%put This log parser is running as SAS Process ID &sysjobid;
%put ******************************************************;
%put ******************************************************;
%put ******************************************************;
options source;

%global AnomalyFound;
%global AnomalyCount;

%let AnomalyFound = N;

libname AnomHist "&FolderForAnomalyHistory";

data AnomHist.Anom_&DT;
length ParseDateTime $ 12 Anomaly $ 60 LogText $ 256 AnomalyCount 8;
retain ParseDateTime "&ParseDateTime" AnomalyCount 0;
infile "&ApplPgmLog" lrecl=256 pad end=LastLine;
input LogLine $ 256.;
Anomaly = ' ';
if LogLine =: 'WARNING' /* BELOW, ignore harmless WARNING messages
                           and harmless NOTEs about irregular
                           situations of no concern to you */
  AND
  (
   indexw(LogLine,'The option expects that the column not contain
all') NE 0 /* Why did I check for this? */
    OR
   LogLine =: 'WARNING: Unable to copy SASUSER registry to WORK
registry.'
   /* Happened during testing with no significant consequences. */
    OR
   LogLine =: 'WARNING: Your system is scheduled to expire'
   /* Happened during testing with no significant consequences. */
    OR
   (LogLine =: 'WARNING:'
      AND
    (indexw(LogLine,'will be expiring') NE 0
       OR
     indexw(LogLine,'expiration') NE 0
       OR
     indexw(LogLine,'SETINIT') NE 0
       OR
     indexw(LogLine,'warning period',' .') NE 0)) /* This is part of
```

```
        added WARNING message lines associated with the one above. */
  ) /* close of AND conditions */
then delete;
else
if LogLine =: 'ERROR: Errors printed'
then delete;
else
if LogLine =: 'ERROR:' /* This detects any ERROR, that is not flagged
                          as a NOTE containing the string _ERROR_ */
then do;
  Anomaly = 'ERROR';
  LogText = LogLine;
end;
else /* Do this check next. It is an IMPORTANT WARNING */
if indexw(LogLine,'not resolved.') NE 0
then do;
  Anomaly = 'macro variable not resolved';
  LogText = LogLine;
end;
else
if indexw(LogLine,'has never been referenced.') NE 0
  and
   indexw(LogLine,'variable') NE 0
then do;
  Anomaly = 'some variable was never referenced';
  LogText = LogLine;
end;
else /* This detects any other WARNINGs */
if LogLine =: 'WARNING:'
then do;
  Anomaly = 'WARNING';
  LogText = LogLine;
end;
else /* The remaining checks are for NOTEs, which might be for
        a problem or situation counter to your intentions */
if indexw(LogLine,'uninitialized',' .') NE 0
then do;
  Anomaly = 'var(s) uninitialized';
  LogText = LogLine;
end;
else
if indexw(LogLine,'too small') NE 0
then do;
  Anomaly = 'format too small';
  LogText = LogLine;
end;
else
if indexw(LogLine,'too long') NE 0
then do;
  Anomaly = 'TITLE or FOOTNOTE too long';
  LogText = LogLine;
end;
```

```
else
if indexw(LogLine,'MERGE statement has more than one data set with
repeats of BY values.') NE 0
then do;
  Anomaly = 'non-distinct keys both sides of merge?';
  LogText = LogLine;
end;
else
if indexw(LogLine,'Division by zero') NE 0
then do;
  Anomaly = 'Division by zero';
  LogText = LogLine;
end;
else
if indexw(LogLine,'_ERROR_') NE 0
then do;
  Anomaly = '_ERROR_ (i.e., data error)';
  LogText = LogLine;
end;
if Anomaly NE ' ' then do;
  output;
  AnomalyCount + 1;
  call symput('AnomalyFound','Y');
end;
if LastLine
then call symput('AnomalyCount',trim(left(AnomalyCount)));
run;

data _null_;
length PID $ 16;
set StorePID.PID_&DT;
call symput('LaunchedProcessID',trim(left(PID)));
run;

%put &LaunchedProcessID is &LaunchedProcessID;

options nocenter;

%if &AnomalyFound EQ Y
%then %do;

proc sort data=AnomHist.Anom_&DT(keep=LogLine)
out=work.DistinctAnomalies nodupkey;
by LogLine;
run;

ods noresults;
ods _all_ close;
ods html path="&FolderForAnomalyHistory" (url=none) style=minimal
         body="Summary Of Anomalies for Program &Program Run by SAS
Process &LaunchedProcessID at &DT..html"
```

```
      (title="Summary Of Anomalies for Program &Program Run by SAS
Process &LaunchedProcessID at &DT");

title font='Albany AMT/Bold'
"Distinct Anomalies in SAS Log for Program &Program Run by SAS Process
&LaunchedProcessID at &DT";
proc print data=DistinctAnomalies noobs;
var LogLine;
run;

title  font='Albany AMT/Bold'
"Counts of Anomalies By Type in SAS Log for Program &Program Run by
SAS Process &LaunchedProcessID at &DT";
proc freq data=AnomHist.Anom_&DT;
tables Anomaly / list nopercent missing missprint;
run;

ods html close;
ods listing;

%let SubjectSuffix = Unexpected Messages in SAS Log;

%end;

%else %let SubjectSuffix = No Unexpected Messages in SAS Log;

filename anyname email
         subject="SAS Process &LaunchedProcessID Ended with
&SubjectSuffix"
            from="&FROMemail"
          sender="&FROMemail"
              to=&Notify /* allow for multiple email addresses */
%if %length(&CCemail) NE 0
%then %do;
              cc=&CCemail
%end;
%if %length(&BCCemail) NE 0
%then %do;
             bcc=&BCCemail
%end;
         ;

data _null_;
file anyname;

put ' ';
put "This message is from SAS Process ID &sysjobid";
put ' ';
put "This email was sent by the log parsing phase of EGbatch.";
put ' ';

%if &AnomalyFound EQ Y
```

```
%then %do;

put "It found &AnomalyCount unexpected messages in the SAS Log for
Program &Program Run by SAS Process &LaunchedProcessID at &DT..";
put ' ';
put "A simple report that identifies the unexpected messages can be
found at:";
put "&FolderForAnomalyHistory.\Summary Of Anomalies for Program
&Program Run by SAS Process &LaunchedProcessID at &DT..html";
put ' ';

%end;

put "If interested, you can find the full SAS Log for Program &Program
Run by SAS Process &LaunchedProcessID at &DT in:";
put "&ApplPgmLog";
put ' ';

%if %length(&SupportPerson) NE 0 %then %do;

put "For more information, please confer with &SupportPerson";
  %if %length(&SUPPORTemail) NE 0 %then %do;
put "via email to &SUPPORTemail";
  %end;
  %else %do;
put ".";
  %end;

%end;

run;

proc printto; run;

%mend EGbatch;
```

ShowProcessID Macro

```
/* Run this macro at the start of your SAS Enterprise Guide session to be
able to distinguish your current SAS Enterprise Guide session's process from
the other one(s) that you want to display and possibly terminate. The
CodeFile you submit with EGbatch will run under a process ID different from
that of your EG session that uses EGbatch. */

%macro ShowProcessID;

%put Process ID for this SAS Enterprise Guide session or SAS batch job is
&sysjobid;

%mend  ShowProcessID;
```

DisplayAllMySASprocesses Macro

```
%macro DisplayAllMySASprocesses;

%global SaveLineSize;
%let SaveLineSize = %sysfunc(getoption(LineSize));
options LineSize=229;
filename TaskList pipe 'tasklist /v';
data _null_;
infile TaskList lrecl=229 pad;
input @1 CommandResponse $char229.;
if _N_ LT 4 then put CommandResponse;
else
if index(CommandResponse,"&sysuserID") NE 0
  and
   CommandResponse =: 'sas.exe'
then put CommandResponse;
run;
options LineSize=&SaveLineSize;

%mend  DisplayAllMySASprocesses;
```

TerminateProcess Macro

```
%macro TerminateProcess(ProcessID=);

data _null_;
cmd = "'tasklist /v /fi " ||'"' || "PID EQ &ProcessID" || '"' || "'";
call symput('taskcmd',trim(left(cmd)));
run;
filename TaskList pipe &taskcmd;
data _null_;
infile TaskList lrecl=229 pad;
input @1 CommandResponse $char229.;
if CommandResponse EQ 'INFO: No tasks are running which match the specified
criteria.'
then do;
  call symput('Found','N');
  put "Process ID &ProcessID was not found.";
end;
else do;
  if _N_ GE 4;
```

```
  if index(CommandResponse,"&sysuserid") EQ 0 then do;
    call symput('Found','N');
    put "Process ID &ProcessID is not for User ID &sysuserid and will not be
killed.";
  end;
  else call symput('Found','Y');
end;
run;

%if &Found EQ N %then %goto MacExit;

data ExamineCommandAndResponse;
cmd = "'taskkill /fi " ||'"' || "PID EQ &ProcessID" || '"' || " /F'";
call symput('taskcmd',trim(left(cmd)));
run;
filename TaskKill pipe &taskcmd;
data _null_;
infile TaskKill lrecl=229 pad;
input @1 CommandResponse $char229.;
put CommandResponse;
run;

%MacExit:

%mend  TerminateProcess;
```