
**Paper SM06**

# Adapting the SAS Enterprise Guide Log Summary to Check Your LOG for You

Lawrence Heaton-Wright, IQVIA, Reading, UK

## ABSTRACT

Checking SAS® LOG Files is something we all do. It's to make sure that the SAS program we've run works as it is expected, to produce the expected results and that there are no messages that violate your company's Good Programming Practice. SAS LOG files can be checked manually as well as in an automated manner by SAS macros, customisable text editors or other types of script files. SAS Enterprise Guide® (and SAS Studio) have the excellent Log Summary window allowing users to click to the relevant Log note.

This paper will give an overview of these processes as well as explaining how to combine the EG Log Summary and a Log-checking SAS macro to produce automated LOG checks as programs are developed which instantly show LOG messages of interest as warnings at code submission time rather than post-production.

## INTRODUCTION

Checking the SAS log information is something every programmer should do. At the most basic level you should check whether the program has any ERRORs. If you have an error then you'll need to investigate what is causing the error and (hopefully) fix the problem. If you have no errors in your log file, then you move to checking whether there are any WARNINGs and deal with those.

Then if you've dealt with all the errors and warnings in your program you may start looking in more detail at the NOTEs that SAS provides. Some companies have Good Programming Practice (GPP) guidelines that do not allow the presence of some of these SAS notes. If you have any of these prohibited notes then you treat them the same as a warning or error and update your program to cope with them. You re-run your program and check the log file again. Hopefully it is now clear of any errors, warnings and specific notes. There are several methods of finding these issues in the log file but most of them are post-execution.

This paper gives an overview of these post-processing methods and explains how to customise the SAS Enterprise Guide (EG) Log Summary so that SAS log notes of interest are promoted to warnings which will appear immediately after executing code within EG which means that you can identify and deal with these issues as you're developing code rather than in a post-processing manner.

## WHY CHECK LOG FILES?

A SAS log file gives you information about whether the program has executed correctly. At the most basic level it's checking whether the program ran without errors. The log file provides us with a wealth of information about syntax issues, timing information, SAS set up (librefs, options, etc.) as well as confirming that the program has run without errors.

Strictly speaking we are not checking that the results produced are correct - we're checking that it's run without errors, warnings or notes of interest that violate our company programming guidelines. Ensuring that the program has produced the correct result is a combination of the programmer checking the log file, checking that the produced results (dataset, table, listing, figure, spreadsheet, xml file, etc.) "makes sense" based on the known input data and by the independent QC process mandated by your company. LOG checking is one part of the whole process. It's an extremely important part and there are quite a lot of aspects of LOG checking that can be automated.

Once the issues have been found, identify what the issue is in the program, update the program (fixing any other issues you can see in the code), re-run the program and check the log, fix any issues, re-run, etc., etc. This is why checking the LOG file is so important - it gives you the information about what your program has done when it's executed. Once your log file is clean of issues then you can move onto checking the output produced. But the important part is that you must check the log.


#### NOTES OF INTEREST

Surely all SAS LOG notes are of interest? They are but some are more interesting than others. When SAS detects a division by zero in a calculation it doesn't show an error or warning but it displays a note in the LOG file. When SAS successfully assigns a libref a note appears in the LOG explaining the libref has been assigned.

If I'm checking that the program is executing as expected and is performing calculations successfully I'm not really interested in if a libref is successfully assigned as my project set up file should handle this side of things and if it hasn't there will be an error in the LOG.

What I'm interested in is seeing that there's a division by zero. But searching for this manually means typing "division by zero" in a search box along with other LOG notes like this. This is plainly an inefficient process which is entirely reliant of the user typing in search terms with the possibility of spelling mistakes or not remembering all the search terms.

#### LIST OF ITEMS TO CHECK FOR

As SAS doesn't really have error codes (apart from the odd note/error - NOTE 484-185: Format XXXX was not found or could not be loaded.) - which would make LOG parsing a lot easier - we need to define a list of NOTES we want to check for.

This needs to be defined by checking LOG files for SAS "Notes of Interest" and adding them to a set of parameters to search for.

These search parameters need to be reviewed on a regular basis to ensure that required notes are being identified.

Once these Notes of Interest have been identified, this list can be utilised by the automated tool of your choice.

## HOW DO WE CHECK SAS LOG FILES?

SAS Log files can be checked manually using a text editor (like NotePad) and looking through the Log for issues in an automated manner using SAS macros and/or other tools like an expandable text editor such as UltraEdit®. Automating the log checking means identifying all the notes of interest as text strings (because SAS does not have return codes for the notes) and then defining a script to check the LOG file for notes of interest.

#### MANUALLY CHECKING THE LOG

You can manually check the SAS LOG in several ways - this can be switching to LOG window in SAS (PC-SAS, EG, SAS Studio) or opening the LOG file in a text editor of your choice.

You can then search for ERROR, WARNING, uninitialized, etc. Make a note of where the issues are, switch back to your program and (hopefully) fix your code. It can be a tedious, boring and repetitive process and you could miss out on some important notes because the process used is entirely manual (and SAS does produce a lot of information in the LOG).

Having said that, manually checking the LOG is probably the best method for checking logical issues with your code. ERRORs, WARNINGs and NOTEs are (usually) the result of syntax errors. Logical errors are the result of correct syntax but the algorithm/logic is incorrect.

If you have a step in your code that doubles the amount of observations in a dataset then it could be syntactically correct but you could have added an OUTPUT or DELETE statement in the wrong place to create more (or less) records than you were expecting. The only way to check this kind of logic is to check the number of records used as input and compare it to the number of records output to the resultant dataset. This information is available in the LOG file. PROC SQL joins are something that it's worth checking for this kind of information as they can be harder to de-bug than DATA steps.

**AUTOMATED CHECKING**

We've identified that manual checking of SAS log files, whilst possible and necessary for some types of checks, is not really consistent or scalable for checking a lot of LOG files. So this means some form of automated checking is required.

**SAS RETURN CODES**

When SAS batch submits a job it generates a return code. This code can be interrogated to ascertain whether the program has run without WARNINGS or ERRORS. If the job just generates NOTES then the return code is set to zero. If WARNINGs are generated the return code is 1. If ERRORs are generated (up to FATAL errors) then a return code of ≥2 is generated.

This can be used to return some quick information to the user that their program has run but has generated WARNINGS or ERRORS. This can be a nice, simple method for identifying whether programs have executed without errors which gives the user an indication of what files can be checked first (the ones with errors, then warnings, then notes only).

However, just because only NOTES have been generated it doesn't mean the LOG file is clean. This is where checking the LOG for Notes of Interest is needed. Having discarded the manual LOG checking as a systematic method for checking NOTES in the LOG, we can turn our attention to automated methods for analysing the LOG file.

**AUTOMATED USING ANOTHER TOOL**

At IQVIA we have used UltraEdit (UE) for several years as our main enhanced text editor. It already has a programming schema display for SAS programming (with colour-coding similarities to the SAS editor). It also has the ability to program tools (hammers) which allow users to write scripts that UE executes. We have several scripts for batch submitting programs, checking GRID status commands as well as checking the LOG file.

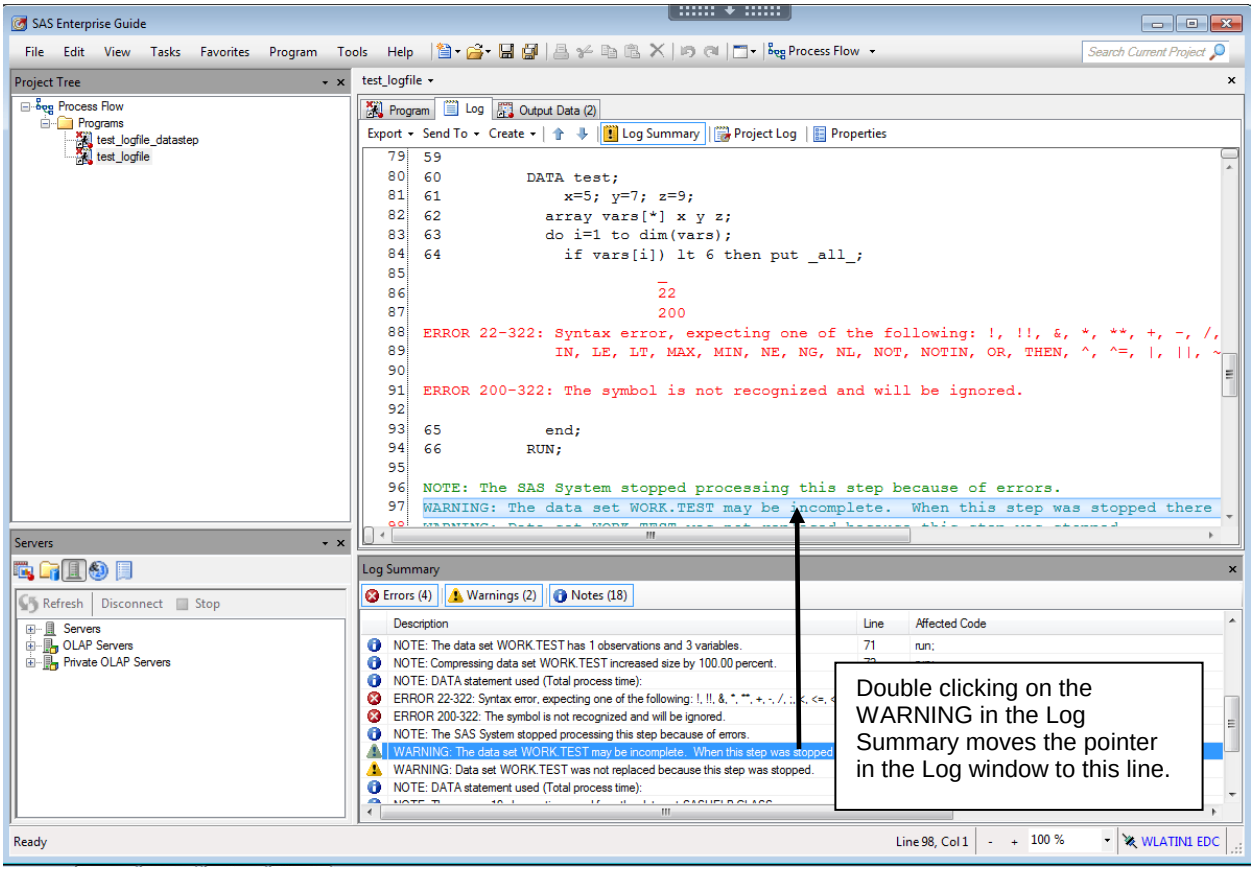
Generally these methods are pretty fast because the application is scanning text files. These scripts need to be carefully written to avoid picking up SAS code rather than the SAS log messages.
The advantage with these is that they can be quicker as they're searching through text files (the LOG) looking for text strings/regular expressions/etc.

**AUTOMATED USING A SAS MACRO**

SAS can, of course, be used to read in the LOG files and scan through the results to identify the Notes of Interest because of the precision that SAS gives us. It can also be used to create reports (say Excel) which give a comprehensive report of the status of all the LOG files in a project. This is probably slower than using another application checking the LOG file (as the LOG file has to be read into SAS and the SAS program steps have to run) but for checking multiple files and the precision of strings to check for it's incredibly useful.

## EG/SAS STUDIO LOG SUMMARY

Both EG and SAS Studio have the useful Log summary tool. This tool gives a summarised view of the number of Errors, Warnings and Notes within the submitted SAS code. This Log Summary also allows you to jump to the actual log message by double-clicking on it or by a right-click option which is much easier than scrolling through the log.




As can be seen the LOG Summary window splits the ERRORs, WARNINGs and NOTEs into 3 grouped views (which look like tabs). This is great for immediately spotting when you have errors and warnings in your log (along with the Program window icon change). What is not so useful is that you will see all NOTEs and this can be 100s/1000s depending on how much code has been submitted.

However the Log Summary is able to jump straight to the point of the LOG directly by clicking on the log summary line. The Log Summary window also allows toggling the visibility of the Notes/Warnings/Errors within the log summary by clicking on the relevant tab. If it's surrounded by a blue line box then this indicates the toggling is turned on and those log messages will be visible within the log summary.

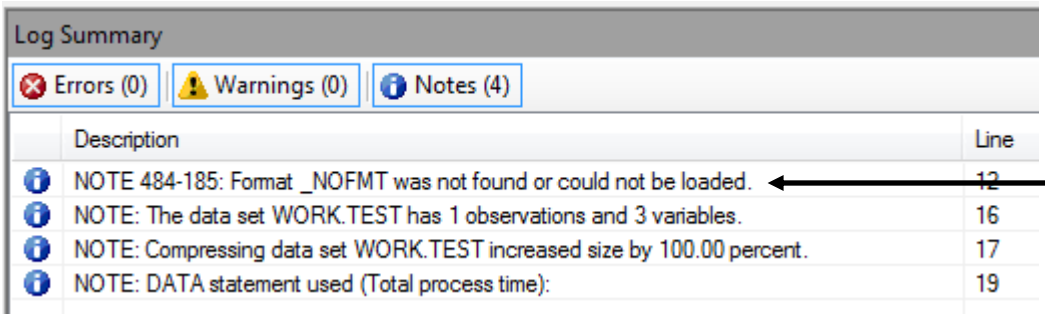
This incredibly useful window combines some of the automation features we would like where ERRORS, WARNINGS and NOTES are summarised and split into 3 groups but we still have the manual process of searching through the NOTES for those of interest. One of the drawbacks of SAS Log files is that the number of NOTEs for most programs can range from 10-20 up to 100s or 1000s for complex programs. This excess of information can be difficult to work with which is why customisation of the Log Summary was investigated.

## CUSTOMISING THE EG LOG SUMMARY

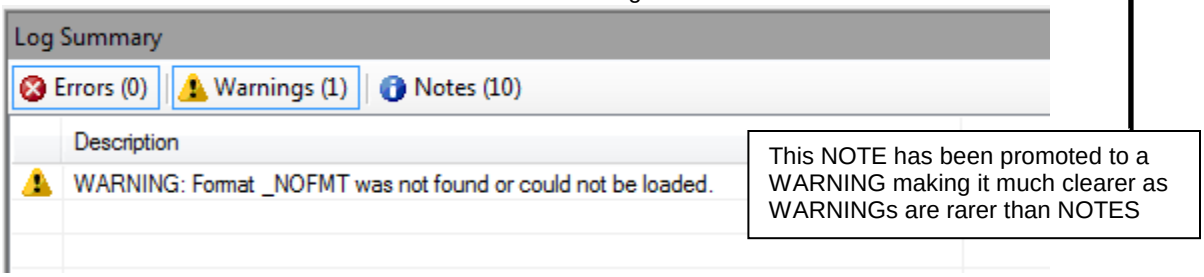
At IQVIA we migrated to a SAS Grid environment and adopted EG as our primary programming interface where previously we had used the SAS Display Manager (DMS). The Log Summary window was a leap forward for the reasons explained above. A SAS log checking macro or UltraEdit tool is used to automate the search for notes of interest in SAS log files but these are all dependent on parsing through a LOG file after it has been created.

After some investigation of the options that EG has for executing SAS code before and after submission, we were able to apply the SAS log checking macro algorithm to the log produced by EG and promoting the Notes of Interest to WARNINGs. The Log summary window then parses this modified log and the newly-promoted WARNINGs are shown the warnings tab. This meant that it's much more obvious whether a Note of Interest has been detected.

Without the customisation of the LOG we would typically see something like this:




With the customisation of the LOG we would see something like:



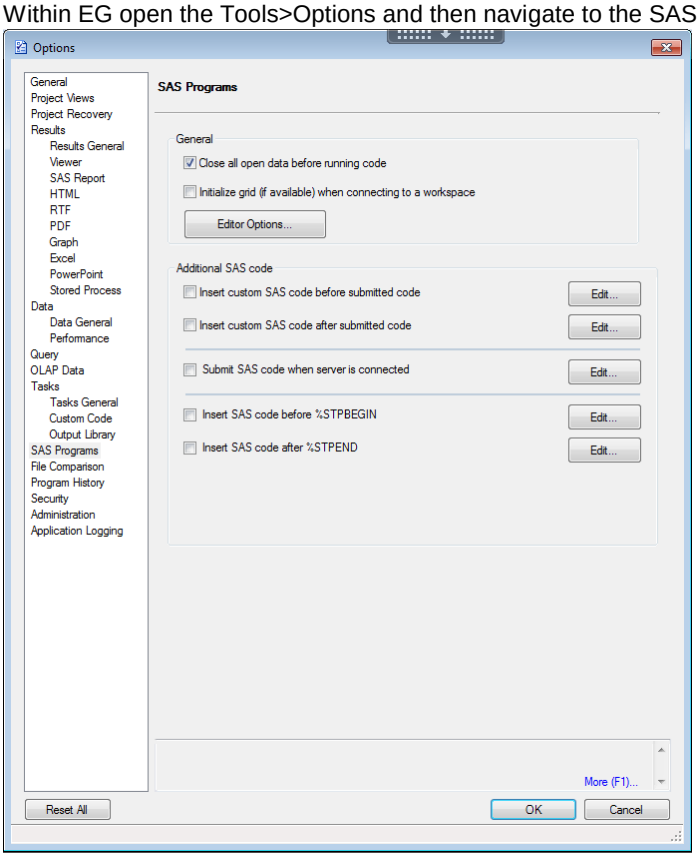

This means that when we submit SAS code, any notes we are interested in will not get lost in the forest of NOTEs. If we consider how many notes that SAS produces even from a few data steps and procedures picking out notes of interest is a slower, manual process.

By promoting them to a WARNING, then we will (hopefully) have a much smaller list to scan through (the Warnings tab rather than the Notes tab). By toggling the Notes grouping tab to not be visible then we will just have Errors, Warnings and promoted Notes.

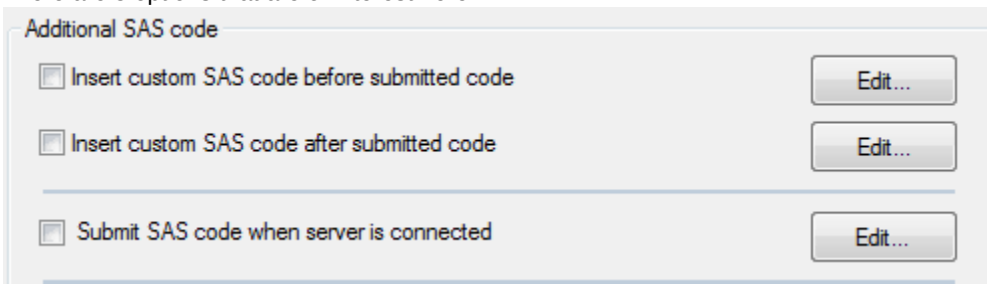
The SAS code used to perform this task is essentially the same as the SAS macro approach to parsing LOG files but with the bonus of happening at development time. This should mean that it's easier to deal with these LOG issues at development time rather than after running the whole program and checking after the program has submitted.

**EG OPTIONS TO USE**

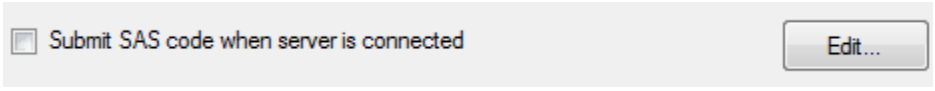
Within EG open the Tools>Options and then navigate to the SAS Programs options:



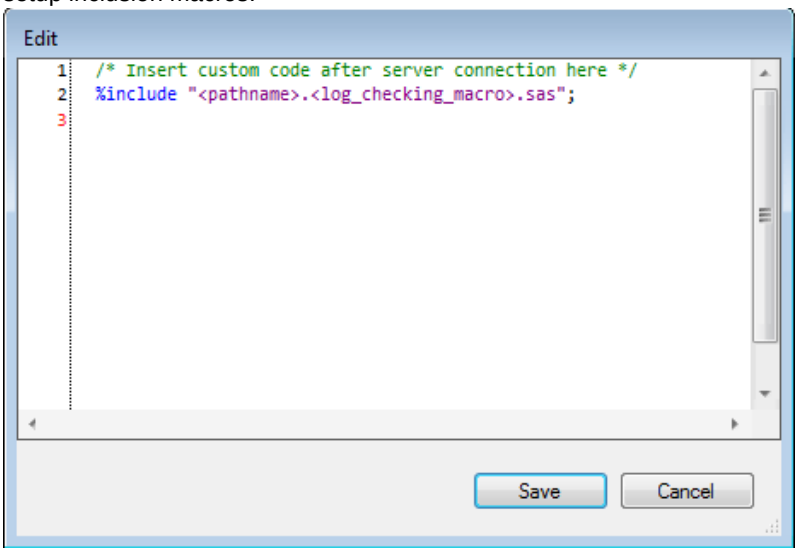

There are 3 options that are of interest here:



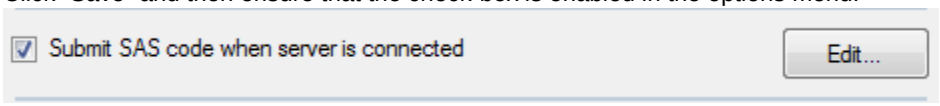
**SUBMIT SAS CODE WHEN SERVER IS CONNECTED**




This option allows us to submit SAS code after EG is opened and a connection to the SAS server is first enabled. In the edit SAS code box, %INCLUDE the macro that is required to enable LOG scanning as well as any global project setup inclusion macros.



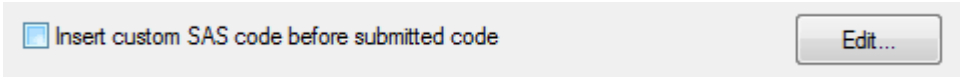

Click “Save” and then ensure that the check-box is enabled in the options menu:



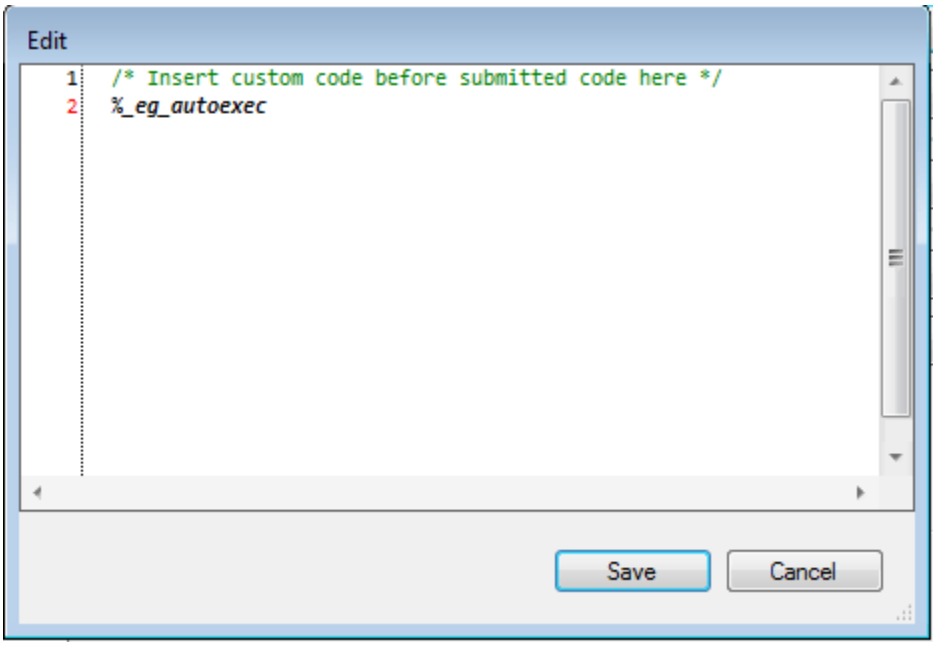

At IQVIA we have a standard set up file process for each project which defines global options, librefs, filerefs, macro locations, etc. When we open EG we want to be able to take advantage of this automation so that we can operate in EG as we do when we submit via SASGSUB.

This code, alongside the LOG checking macro code, is %INCLUDEd at the server connection time as we only want to include and compile this macro code once. A %INCLUDE is used in place of defining a SASAUTOS path or a link to a SAS macro catalog because these can be changed/overwritten by different study set up OPTIONS statements. By using a %INCLUDE the SAS macro is compiled and added to the WORK macro catalog.

**INSERT CUSTOM SAS CODE BEFORE SUBMITTED CODE**




This option allows us to submit SAS code before the actual SAS code we’ve written (in much the same manner as the EG wrapper code). By clicking on the Edit button we get a similar window to the previous option where we can add the macro call.




Click “Save” and then ensure that the check-box is enabled in the options menu:

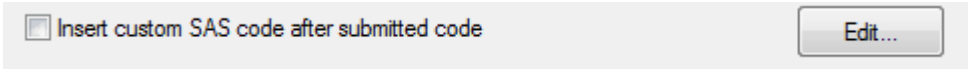

```
%macro _eg_autoexec;
  %*************************************************************************
  %* Turning off macro options
  %*************************************************************************;
  %let __origopts = %sysfunc(getoption(mprint)) %sysfunc(getoption(symbolgen))
                    %sysfunc(getoption(mlogic));
  options nomprint nosymbolgen nomlogic;
  %*************************************************************************
  %* Change log destination prior to submitting code
  %*************************************************************************;
  %* Getting the WORK directory;
  %global __workpath;
  %let __workpath = %sysfunc(pathname(work)); ❶

  proc printto log = "&__workpath.\_EG_TempLog_&sysuserid..log" new; ❷
  run;
  %*************************************************************************
  %* Restore original options
  %*************************************************************************;
  options &__origopts.;
%mend _eg_autoexec;
```

❶ In order to re-route the LOG file, the first thing is to find the actual pathname of the WORK directory. As the WORK library is a physical storage location, we can take advantage of this and use the PATHNAME function to extract this information into a global macro variable (as this needs to be used in subsequent macro calls).

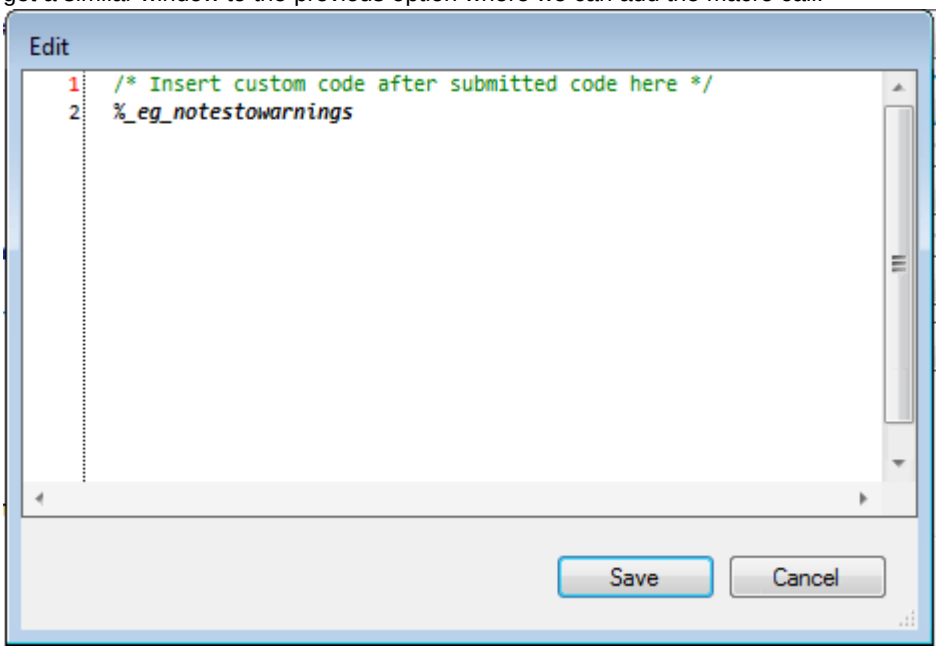
❷ Use this macro variable in a PROC PRINTTO to re-route the LOG file to an external file stored in the WORK folder.

This macro is executed prior to any code submitted within the EG program window. This means that the LOG file for the executed code is sent to the PROC PRINTTO file and not the EG wrapper code.

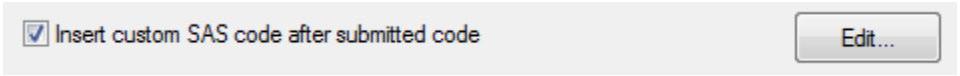
**INSERT CUSTOM SAS CODE AFTER SUBMITTED CODE**




This option allows us to submit SAS code after the actual SAS code we've written. By clicking on the Edit button we get a similar window to the previous option where we can add the macro call.




Click "Save" and then ensure that the check-box is enabled in the options menu:




The _EG_NOTESTOWARNINGS macro (which is %INCLUDEd on server connection) does the actual promotion of Notes to Warnings.

```
%macro _eg_notestowarnings;
  %*************************************************************************
  %* Turning off macro options
  %*************************************************************************;
  %let __origopts = %sysfunc(getoption(mprint)) %sysfunc(getoption(symbolgen))
                    %sysfunc(getoption(mlogic));
  options nomprint nosymbolgen nomlogic;

  %*************************************************************************
  %* Convert notes of interest to warnings after submitting code
  %*************************************************************************;
  proc printto log = log; ❶
  run;

  data _null_; ❷
    length line $32767;
    infile "&__workpath.\_EG_TempLog_&sysuserid..log" length = __length;
    input @1 line $varying32767. __length;

    %* Converting NOTE xxx-xxx to NOTE:; ❸
    if kscan(line, 1, " ") in ("ERROR", "WARNING", "NOTE", "INFO") then do;
      length code $10;
      code = kscan(line, 2, " ");
      if kindex(code,":")=klength(code) and not kverify(kstrip(code),"1234567890-:")
      then line=kscan(line, 1," ")||":"||ksubstr(line, kindex(line, ":")+1); ❹
    end;

    %* Converting NOTE and INFO of interest into warnings;
    if ksubstr(line, 1, 6) in ("NOTE: ","INFO: ") then do; ❺
      if
        kindex(kupcase(line), "UNINITIALIZED") or
        kindex(kupcase(line), "INVALID") or
        kindex(kupcase(line), "DIVISION BY ZERO") or
        ...
        kindex(kupcase(line), "SOME GRAPH LEGENDS HAVE BEEN DROPPED DUE TO SIZE
                               CONSTRAINTS")
      then line = "W" || "ARNING: " || ksubstr(line, 7);

      else if kindex(kupcase(line), "THE DATA SET") or ❻
         kindex(kupcase(line), "NOTE: LIBRARY") or
         kindex(kupcase(line), "NOTE: LIBREF") or
         ...
         kindex(kupcase(line), "NOTE: FILEREF") or
         kindex(kupcase(line), "NOTE: TABLE") or
         kindex(kupcase(line), "NOTE: SQL")
        then line = catx(' ','INFO:',ksubstr(line, 7));
    end;

    %* Print modified log;
    put line; ❼
  run;

  %*************************************************************************
  %* Restore original options
  %*************************************************************************;
  options &__origopts.;
%mend _eg_notestowarnings;
```

The _EG_NOTESTOWARNINGS macro executes after the submitted code has finished executing.

❶ The first step is to close the PROC PRINTTO LOG file that was opened via the _EG_AUTOEXEC macro.

❷ Next a DATA _NULL_ step reads in the LOG file (a DATA _NULL_ is used as we don't want to produce a dataset).

❸ The code used for identifying Notes of Interest is essentially the same code as used in a SAS macro-based approach for parsing LOG files. Each LOG file line is read into a variable and the text is scanned to identify whether it is a SAS-produced LOG note.

❹ Notes of the form NOTE xxx-xxx are converted to "NOTE:" format.

❺ By identifying whether the LOG line starts with "NOTE:" the subsequent text is scanned to determine if there are Notes of Interest. If these are detected, then "NOTE:" is replaced with "WARNING:". Some INFO log notes may also be changed (Overwritten variables for example) so these are flagged as well.

❻ A similar approach to demote notes to information is taken. This allows notes that are simply information (for instance a LIBREF has been assigned) to not be shown in the Notes summary tab.

Therefore, log notes that have not been accounted for (either by promotion to Warnings or demoted to Info) will pop up in the Notes tab highlighting that these have not been accounted for. Such notes can then be included in the _EG_NOTESTOWARNINGS macro for future programs.

❼ The PUT statement at the end of the DATA step will then write all the LOG notes to the LOG window and the EG Log Summary window will surface these to the correct tab within the Log Summary window.

**EG LOG SUMMARY CUSTOMISATION**

By utilising these 3 EG options and by using the macro code shown above, the EG Log Summary can be customised to help programmers identify Notes of interest when developing SAS programs. Identifying these issues at run time can help to reduce the cyclical process of running programs, checking the log files, updating the programs, re-running, checking the log files, etc.

It should be noted that this doesn't replace the need for post-processing log checks after batch submission but is designed to help the programmer identify and deal with LOG issues when writing the code.

By expanding the customisation to demote some notes to SAS information notes, this can help to identify less common SAS log notes that have not been accounted for in the _EG_NOTESTOWARNINGS macro. Notes discovered via this method can then be added to Notes of Interest or can be demoted to information notes as appropriate.


## WHAT IS THE BEST LOG CHECKING METHOD?

It's vitally important to check LOG files. It is the only way we have of ensuring that the programs have executed without error and that all the settings defined have been correctly applied and that the program adheres to any GPP in place.

An automated process can pick up syntax errors but logical errors can generally only be picked up by the user (for instance doubling the number of records in a dataset is perfectly acceptable code with no syntax errors but might not be the logic you require).

- Automation using another tool is great for rapid checking of log files
- The SAS macro method is especially useful when checking a whole set of LOG files to exacting standards
    - It is part of a lead programmer's role is to make sure the programs are following programming standards, no errors or warning are appearing, etc.
- The return code methodology is useful when combined with a batch submission script to give a quick indication that all programs have run without SAS generated ERRORs or WARNINGs
- Manually checking a LOG file is the main method for finding logical errors and issues

When developing SAS programs, modifying the EG Log Summary method combines the best of both worlds in that the existing technology (the EG Log Summary) is customised with our log checking requirements for Notes of Interest without losing the ERROR and WARNING notes that SAS already provides in an automated manner.

This process of promoting Notes of Interest to warnings highlights these issues to the programmer instantly. By demoting other notes to information, SAS Log Notes that have not been accounted for in the Note of Interest are highlighted as Notes rather than being buried in a massive section of SAS log notes.

The programmer can then deal with such issues when developing their code which is always the best time to handle these problems. This method also allows users to manually check their LOG as well so that logical issues can be picked up as none of the notes are lost and will still appear in the main Log tab.

As discussed there are several methods that can be used to check LOG files. Each of the methods presented above have their place. It's a case of picking the best method for what you're doing at the time.

## CONTACT INFORMATION

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

Lawrence Heaton-Wright
IQVIA
3 Forbury Place,
23 Forbury Road,
Reading,
RG1 3JH,
United Kingdom

Work Phone: +44 (0) 118 450 8320
Email: lawrence.heaton-wright@iqvia.com
Web: https://www.iqvia.com/