SAS® LOG/LST Filename improvement for easier: code review, code audit and project development history and chronology.

Zeke Torres Code629 Chicago, IL

ABSTRACT

The SAS® config file is powerful and useful. In this example, we show how to customize the names of the Log and List files from the code we run, and obtain a useful new name as the result.

> The new name of a log file might look like this:

```
20180904_hhmm_userID_name-of-code-that-was-run.log.
```

The benefit is that when one user or a team of users builds the code, everyone can see its progress. In this way, collaborating is simplified and results are easier for colleagues to share and consume. This is a mock or proposed file name. This macro is intended to inspire you to see what additional SAS System values to add to your style.

INTRODUCTION

The typical problem encountered is we can run our SAS code and get useful output: LOG/LST (List) – but after we run it again – that original output is gone. Its overwritten.

Or if one of our co-workers or team members runs the same code – again – now our LOG/LST become harder to pin down – who – ran that code.

This is a solution meant to utilize the SAS Configuration file and a set of SAS Macros to improve the name of the LOG/LST output.

There are already ways to customize those – with SAS options. I will quickly cover those here. But this is more of a recommended methodology for you to consider. My reason for employing this technique is because often I am searching for information in many LOG/LST. Either to debug, help someone debug (Myself, Team Member, Client) or to track progress on a "Project/Build" of work that is critical to gauge where and how things are. Especially when one or more work on something like this.

I will mention the use of GIT or additional methods to consider on how to organize your code. But those are not the focus of how to obtain and enable the improved path/naming of LOG/LST. But knowing GIT is not a requirement of this paper.

HOW TO QUICKLY IMPLEMENT – PROOF OF CONCEPT

Quickest way to this Solution: See the Appendix, Copy Code, run code - go!

OVERVIEW OF REQUIREMENTS

USER SKILL LEVEL REQUIREMENTS

The intended reader and user should have a basic or better comprehension of:

- SAS Macro Language
- Operating System Basics and User Permissions
- SAS Config File
- PDV

- Data _null_
- SYSPARMS
- Data Used for this Paper/Topic

Extensive knowledge of these items is not a pre-requisite. Great – but not needed.

The paper should provide a useful template for someone to observe those elements working together. Being familiar with those elements is simply going to determine how quickly and easily the reader/user can take this code configure it to work and get it customized to their situational needs.

SAS MACRO LANGUAGE SKILLS

There are uses of SAS Macro language. The code I've crafted works well as a macro and is called in during compile. A reader/user of this paper/code should have an intermediate to advanced understanding of SAS Macros. A reader with less knowledge should still be able to use this method and code – but I highly recommend using the References section to improve your SAS Macro Language skills before engaging in this type of overall technique.

OS BASICS AND USER PERMISSIONS

This method works on Windows/Linux/Unix – however – each of those environments brings with it overall considerations on "where" you can locate your SAS Config. As well as how or where you can actually execute this macro to bring about the desired results.

This means – that if your SAS environment is highly regimented and secure – you could still get some of this to work. But not as much as say – if you had better "Admin" privileges. Another consideration for readers – this is ideal when you can work with your IT/Admin and implement it across your team. Reducing the need for individual users to modify or change SAS config etc.

The reader of this paper should understand that SAS Config files and updating should be done after back up copies have been made.

SAS CONFIG FILE

I will give brief explanations on how to locate your SAS Config file. The References section will also cover some papers by other authors that describe this in more/better detail.

The important part of this paper and SAS Config is to allow you (the user) to tailor a set of options that are available when SAS compiles code. Once SAS starts to run the code -we've lost the chance to use those options. In my opinion – the careful and strategic use of the SAS Config options at compile – is one of the many ways coding in SAS is really powerful.

This paper does not do the SAS Config and Compile topic justice.

I encourage you to become more familiar with those parts of SAS.

DATA _NULL_

We will use Data _NULL_ and create macro variables. Then recreate or re-use those macro variables and values. It would be useful for a reader to be familiar with SAS Macro Language and the Data _NULL_ topic – but its not a prerequisite. The use and implementation of this code will provide a reader with a good example of how this works.

SYSPARMS

SAS Sysparms will be used by this process and code. We will capture the values from those fields within the SAS System/Session and revise, repurpose them to suite our needs.

DATA USED FOR THIS PAPER/TOPIC

We don't have heavy requirements for data here. The data I've used in the paper will come from "SASHELP" libraries.

BEFORE – THE PROBLEM
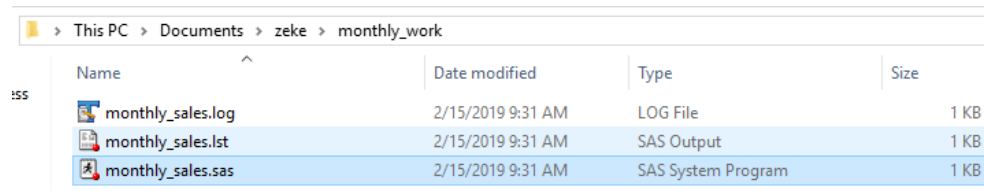
The scenario is: One user has a SAS job.

Its kept by the user in a folder – and s/he is the only person on the team.

> For this section of the SAS paper - Its called "`monthly_sales.sas`".

> When this job is run – it will produce:

"`monthly_sales.log`"

"`monthly_sales.lst`"

SAS generates the name of the LOG/LST automatically during compile based on the (dot)sas code file name. Which is typically useful.

Right now we know good useful facts – the OS is informing us:

- When it was run – We have a Date/Time Modified (here from Windows)
- We know "who" because we know its ONE person (us).
- We know the job.

That's about it useful – for now but vulnerable. What if our SAS work is evolving and being updated, changed as we go. What if the code for this topic needs to be run more than once:

- Once for one region **again** for a different region?
- Or **again _after_** totals or data from the source are complete.
- Any other reason – including debugging or optimization.

PROBLEMS COME UP QUICK

What results is the loss of useful details. First – when we "rerun" the logs and list we currently have – will be over-written. Gone.

What if I'm trying to evaluate its performance and compare?

I would have to go to this location and 'rename' these (dot)log and (dot)lst. Thereby protecting them. It can get very annoying – quickly.

What if for data refreshing purposes the first and subsequent times its run – are equally valid? Well I would loose any history or reference of a successful process with each pass.

This problem – becomes that much more severe once we expand the team to more than one person.

RESULTS OF NEW NAMING

Before we had one person working on that code. Now we have more.

This example assumes we have gathered the logs and list files and code together.

Its meant to demonstrate the concept of the file name method.

In this scenario ASMITH ran the code on 2/13 at 3:47pm.   S/He found an issue and called for help to solve it. ZTORRES ran some tests on 2/13 at 6:47pm. The two solved the problem.

We can then see the code was again run by both on 2/15 and the relevant details.

> This PC > Documents > zeke > monthly_work			
Name	Date modi...	Type	Size
20190213_1547_asmith_PROD_monthly_sales_east.log	2/15/2019 ...	LOG File	1 KB
20190213_1547_asmith_PROD_monthly_sales_east.lst	2/15/2019 ...	SAS Output	1 KB
20190213_1547_asmith_PROD_monthly_sales_east.sas	2/15/2019 ...	SAS System Program	1 KB
20190213_1847_ztorres_TEST_monthly_sales.log	2/15/2019 ...	LOG File	1 KB
20190213_1847_ztorres_TEST_monthly_sales.lst	2/15/2019 ...	SAS Output	1 KB
20190213_1847_ztorres_TEST_monthly_sales.sas	2/15/2019 ...	SAS System Program	1 KB
20190215_1004_asmith_PROD_monthly_sales.log	2/15/2019 ...	LOG File	1 KB
20190215_1004_asmith_PROD_monthly_sales.lst	2/15/2019 ...	SAS Output	1 KB
20190215_1004_asmith_PROD_monthly_sales.sas	2/15/2019 ...	SAS System Program	1 KB
20190215_1047_ztorres_PROD_monthly_sales_east.log	2/15/2019 ...	LOG File	1 KB
20190215_1047_ztorres_PROD_monthly_sales_east.lst	2/15/2019 ...	SAS Output	1 KB
20190215_1047_ztorres_PROD_monthly_sales_east.sas	2/15/2019 ...	SAS System Program	1 KB
20190215_1234_ztorres_PROD_monthly_sales_west.log	2/15/2019 ...	LOG File	1 KB
20190215_1234_ztorres_PROD_monthly_sales_west.lst	2/15/2019 ...	SAS Output	1 KB
20190215_1234_ztorres_PROD_monthly_sales_west.sas	2/15/2019 ...	SAS System Program	1 KB

Just by using this method we can obtain useful facts.

We can learn "who" ran the code. Was it for "test" or "production"?

Those facts are on the – left – of the name of the code.

What was the "code" they ran? We want to keep that.

The items to the right – are facts about what was "in" the code we wish to display at this high level. So in the example of jobs run on 2/15 – we can see some were by region. In this example: West, East.

BENEFITS

We can now get more details about work, from any number of users.

In a chronological way. Informative about relevant elements of the work being done.

We can easily track down what log/lst to explore if an issue was reported by someone.

We no longer have to manually go in and rename log/lst – or even remember to do that.

CODE OVERVIEW

Before our program is executed or run. We have places where we can customize options.

Typical places to do that are: Config, Autoexec and even in the code itself – after its started.

But the earliest place we can adjust those options are at compile – the greatest available options are during the SAS engine itself compiling the (dot)sas code we are running.

Its at that stage that the options we wish to modify get set.

If we try to modify the print/log file names from sas within the code – we cant fully adjust to suite our needs. Its too late in the compile/system options process.

The options SAS provides with "PRINTTO" are simply too far along the compiler process.
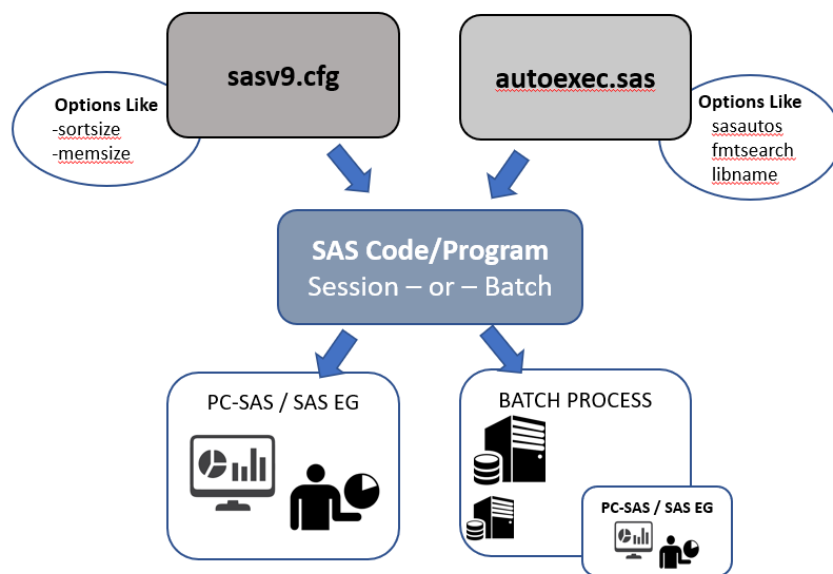
MACRO CODE LOCATION AND PLACEMENT

This macro needs to be located somewhere that you can access and run it. It could be in the directory or folder you are working from. Or you could store it in a catalog or other remote folder.

But it must be called in before SAS actually executes code.

See Attachment 2

The code is ideal for Windows/Linux Batch processing.
One item to be aware of is the "Slashes" and the differences between: Windows/Unix
You may need to make minor configuration to the macro where I've made comments about this.

You may also want to find the SAS executable for your location.
In the example for this document I've taken a Windows variation. The exact location of your SAS.EXE will be something to confirm and verify.

EXECUTION OF MACRO

Here are the basic – long version of the execution parms to get full use of the macro.
For this section of the paper my SAS code is called: "try.sas"
On Command line you will run:

```
>"C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -SYSIN try.sas -autoexec
"zt_log_list_prep_3139.sas" -sysparm "prod"
```

The "SYSPARM" value can be adjusted to include facts we want in revised file name.

```
C:\Users\ETorres\Documents\cmder
λ "C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -SYSIN try.sas -autoexec "zt_log_list_prep_3139.sas" -sysparm "prod_zt_"|
```

In this example: I've set my "SYSPARM" value to "PROD_ZT".
This will allow the output to return a more useful value as part of the filename.

Notice how we "AUTOEXEC" the macro that updates the log/list.
We need to do this at this stage so that all the correct and updated/revised values for PROC PRINTTO (LOG/LST) are available as early as possible in the compile stage.

My suggestion are as follows:
1. Explore this macro by using other values in SYSPARM:
   a. Like: PROD_OTHERUSER, PROD_SSMITH, PROD_MonthlyREport_ etc.
   b. Or: CR_SOMEUSER, TEST_MonthlyReport_ etc.
   c. This macro is set to two variations: PROD and CR (code review)
   d. This way I Know if something was done for production or code review.
2. If you put NO value in SYSPARM – then the macro does not execute.
3. The use of SYSPARM with: PROD, CR as prefix enables it.
4. You will observe that for a SAS code named: "try.sas" the following occurs.
   a. The true TRY.SAS log – still comes up – try.log
   b. The "try.log" is NOT the full log – its only the log up and thru where the PROC PRINTTO was initiated with the new values.
   c. The rest of the List/log AFTER the PROC PRINTTO – are under the new name.

In my own work flow I've enabled more "SAS SYSTEM" features.
Like: UserId's and others. My goal with the paper was to demonstrate the process and revised naming convention that is possible and why I find it useful.

I encourage you to explore adding other SAS System variables as you desire.

What I typically do with my work in my local folders is copy or push the logs and list over to a team folder. But I don't have any issues about filenames or history because I've got a more useful set of log/list filenames.

CONCLUSION

This method of creating logs and lists, storing them and using a standardization method to keep team work easier – is my recommendation. I've employed this method in places where even if I am working as an individual – it still helps me keep track of the progress of work.

THE FINE PRINT

This software is provided to you "as is" without any warranties, express or implied, including but not limited to implied warranties or merchantability and/or fitness for a particular purpose. The Author, SAS Institute and its licensor(s) disclaim any liability connected with the use of the software and/or proposed code solution presented. The Author, SAS Institute offers no technical support for the software and/or proposed code solution presented.

I recommend that you first make useful copies of your code, config and data before engaging in modifications of those in your environment.

BUT WAIT…

After all that in "the fine print" if you still want to exchange some ideas, comment, critique or ask questions – just contact me and lets see what we can figure out together.

REFERENCES

https://www.lexjansen.com/nesug/nesug11/ds/ds04.pdf

https://communities.sas.com/t5/SAS-Communities-Library/The-SAS-Supervisor/ta-p/429216

ACKNOWLEDGMENTS

I want to thank some awesome co-workers that tolerated my numerous attempts to show them this code and equally tolerated me getting it to work easy enough for anyone to use.

Kay Whitman	MPA Healthcare
Bryn David	NORC
Bartosz Jabłoński	Citibank Europe
Kean Chew	Kemper Insurance
Allan Bowe	sasensei.com

RECOMMENDED READING

- *Base SAS® Procedures Guide*

- *SAS® For Dummies®*

CONTACT INFORMATION

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

Zeke Torres
CEO
Code629
Chicago Illinois USA

Email:	info@ztorres.net
LinkedIn:	https://www.linkedin.com/in/zeketorres/
Local SAS User Group:	www.wcsug.com

ABOUT THE AUTHOR

I am a data engineer. The roles I play are preparing and constructing complex data for analytic and statistical use. I enjoy designing and building data driven decision support systems. A typical role for me is that of: data management, reporting and complex analytics. Including the governance of metadata.

I translate complex business definitions, requirements and convert them into algorithms and code. Then tackle the some of the toughest, ugliest most unstructured data (lots of it) and make it valuable and useful.

Crafting end to end "Data to Decision" solutions that matter to an organization. I bridge the gap between the IT DBA's and Statistics Quantitative Analysts. My main tool of choice is SAS®. I'm also learning Python and enjoying it.

One of my hobbies are Local SAS User Groups. I lead the Chicago Area SAS Users Group – WCSUG.com – if you are ever in the area stop by and say hello.

TRADEMARK CITATIONS

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.

APPENDIX 1: CODE

```
/**************************************************************
* PROGRAM NAME :  log_list_prep.sas
* DESCRIPTION :   prepare and modify log and list
* AUTHOR:         zeke torres
* LinkedIn:       https://www.linkedin.com/in/zeketorres/
**************************************************************/


/* obtain the path and number of sections of that path
*****************************************************/
data _null_;
    cpy_sys_var1 ="%sysfunc(getoption(SYSIN))";
    call symput('cpy_sys_path'  ,cpy_sys_var1);
run;

/* pick up the sysparm value from the command line this enables
*  the rest of the code to be OFF by default and only if a user
*  enters the correct value below will the log/list file be copied
*  to the project folder
**************************************************************/
data _null_;
    prod="&sysparm";
    prod =lowcase(prod);
    prod =compress(prod);
    prod_key=0;
    /*** we want to create two types of namig conventions – prod
     *** and CR (code review) for prod or CR – we simply need the
     *** leading values in sysparm to match
     **********************************************************/
    if prod in :('yes','live','prod','production','y','1') then do;
       prod_key=1;
    end;
    if prod in :('code','codereview','cr','2') then do;
       prod_key=2;
    end;
    call symput('cpy_sys_prod',prod_key);
run;


/*** we are going to prep and clean up the sas code path and
***  sas code execution .sas ***/
data _null_;
      /** get the file name from the path ***/
      length temp_file_name  $700.;
      length temp_file_name2 $700.;
      length code_name       $250.;
      temp_file_name ="&cpy_sys_path";

      /** need to reverse to isolate the actual name ***/
      temp_file_name_r=reverse(temp_file_name);
      temp_file_name2 =scan(temp_file_name_r,1,'\');  /** you may need to
adjust the back/forward slash to your os **/

      /** reversing back to final output name ***/
      temp_file_name2=reverse(temp_file_name2);
      temp_file_name2=left(temp_file_name2);
```

```sas
        temp_file_name2=compress(temp_file_name2);
        temp_file_name2=trimn(temp_file_name2);
        xlen=length(temp_file_name2);
        code_name=substr(temp_file_name2,1,250);
        code_name=scan(code_name,1,'.');

        zalen=(length(temp_file_name)) - xlen;

        fnl_path=substr(temp_file_name,1,zalen);
        fnl_path=compress(fnl_path);
        fnl_path=compbl(fnl_path);
        call symputx('cpy_fnl_path',fnl_path);
        call symputx('cpy_code_name_x',code_name);
run;


%macro dnull(job_class=);

/*** we will bring together the facts we need and append to
 ***  revised log and list names ***/
        data _null_;
            /* get the system date and set format for
             * it as yyyymmdd10
             ***********************************/
            length tdate 8.;
            format tdate yymmdd10.;
            tdate="&sysdate9"d;
            length fdate $20.;
            fdate=tdate;
            fdate=put(tdate, yymmdd10.);
            fdate=tranwrd(fdate,"-","_");
            fdate=compress(fdate,"_");

            /* get the system time and compress
            ************************/
            length stime $15.;
            stime="&systime";
            stime=compress(stime,":");

            /* prepare the new log and list names using the new
            * prefix code name now add that to the final string
            * of chars to get a useable set of file names
            * I am using a long var width to ensure I capture all byte
            * values.  This is not intended to hold data ever
            * beyond the null step and macro var
            ***********************************/
            length code_list $3500.;
            length code_log  $3500.;
            cpath="&cpy_fnl_path";
            cfile="&cpy_code_name_x";
            csval="&sysparm";
            code_list ="'" !!  cpath !! fdate !! "_" !! stime !! "_" !!
csval  !! "_" !! cfile  !! ".lst" !!    "'";
            code_log  ="'" !!  cpath !! fdate !! "_" !! stime !! "_" !!
csval  !! "_" !! cfile  !! ".log" !!    "'";
            code_list =compress(code_list);
            code_log  =compress(code_log);
```

10

```
              call symputx('cpy_sys_list',code_list);
              call symputx('cpy_sys_log' ,code_log);
        run;

        /* execute the new file names for both log and list */
        proc printto print = &cpy_sys_list  new;
        run;

        proc printto   log = &cpy_sys_log   new;
        run;
%mend;   *** end dnull macro;


%macro loopy();
    %if &cpy_sys_prod. =1 %then %do;
       %dnull(job_class= );
    %end;  /** end of the criteria if this is prod or not **/

    %if &cpy_sys_prod. =2 %then %do;
       %dnull(job_class=_CR);
    %end;  /** end of the criteria if this is prod or not **/
%mend;

%loopy();
```

APPENDIX 2: HOW TO EXECUTE

On Command line:

> "C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -SYSIN try.sas -autoexec
> "zt_log_list_prep_3139.sas" -sysparm "prod"

The "SYSPARM" value can be adjusted to include facts we want in revised file name.

```
C:\Users\ETorres\Documents\cmder
λ "C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -SYSIN try.sas -autoexec "zt_log_list_prep_3139.sas" -sysparm "prod_zt_"
```