0825-2017

A Browser-Based Tool for Automating SAS Batch Program Generation; How to Get Individual Logs/Outputs

Mohammad-Reza Rezai, Mahmoud Azimaee, Jiming Fang, and Jason Chai-Onn,

Institute for Clinical Evaluative Sciences, Toronto, Canada

ABSTRACT

Migration to a SAS® Grid computing environment provides many advantages. However, such migration may not be free from challenges especially considering users' pre-migration routines and programming practices. While SAS® provides good graphical user interface solutions (e.g. SAS Enterprise Guide®) to develop and submit the SAS® code to SAS® Grid, some situations may need command-line batch submission of a group of related SAS® programs in a particular sequence. Saving individual log and output files for each program may also be a favorite routine in many organizations.

SAS® has provided SAS® Grid Manager Client Utility, and SASGSUB commands to enable command-line submission of SAS® programs to the grid. However, submitting a sequence of SAS® programs in a conventional batch program style and getting individual logs and outputs for them needs a customized approach. This paper presents such an approach. In addition, an HTML/JavaScript tool developed in-house is introduced. This tool automates the generation of a SAS® program which almost emulates a conventional scenario of submitting a batch program in a command-line shell using SASGSUB commands.

INTRODUCTION

Use of SAS® Grid computing is getting more popular with rapid growth of data analytic demands, use of big data and escalating complexity of data access right management in shared multi-user environments. In a SAS® Grid environment, the SAS® computation workload is distributed across a computer cluster on a network which is controlled by **SAS® Grid Manager**®. The users can interact with the SAS® Grid/cluster via a network using their own terminals running a client-side SAS® product like SAS® Enterprise Guide® (SAS® EG) or SAS® Display Manager®. However, it may occasionally be necessary to submit jobs that take a very long time (e.g. days) to run beyond user's allowed active session time or normal working hours. For example, in some shared multi-user organizations, the users may not be allowed to leave a SAS® Enterprise Guide® session running more than certain duration of time (e.g. 72 hrs) - to enable balanced sharing of system resources. Therefore, running very long jobs using SAS® Enterprise Guide® or SAS® Display Manager® may not often be practical and the users may prefer to submit a SAS® program or a sequence of SAS® programs in batch-mode to be processed on grid machines without requiring them to keep logged on their terminal workstations or even without requiring the users to have SAS® installed on their local machines.

Another scenario that may necessitate batch-mode submission of a SAS® program is a program expected to produce very large log or results outputs that may overwhelm a SAS® Enterprise Guide® session probably by creating large embedded outputs in the SAS® .egp file or alternatively due to memory shortage on the client's side. In this settings, the user may also prefer to submit the final program (after testing) via batch-mode and examine the logs, ODS (Output Delivery System) outputs or .lst text files outside SAS® EG.

To meet batch-mode submission of SAS® programs in a SAS® Grid environment, a **command-line** utility called **SAS® Grid Manager Client Utility** have become available. It allows the users to submit a SAS® program for processing on a SAS® Grid without obligation of remaining active or having a SAS® product

installed on their local terminal. SAS® Grid Manager Client Utility has **SASGSUB commands** to handle such submissions.

For example, in our organization, the users work with Microsoft Windows® terminals with SAS® Enterprise Guide® installed locally to interact with the SAS® Grid machines (Figure 1). To take advantage of the above-mentioned batch-mode submission of SAS® programs via SASGSUB commands, a local installation of PuTTY [http://www.chiark.greenend.org.uk/~sgtatham/putty] is used. PuTTY is an open-source and free tool that emulates an **xterm terminal** on Windows platforms. Via PuTTY, the users can run some useful UNIX commands to manage their files on a shared network drive as well as using SASGSUB commands for batch-mode submissions of SAS® programs to SAS® Grid. SASGSUB commands also allow the user to check the status of a job to see if the submitted SAS® program has been processed without error. More details of SASGSUB commands are beyond the scope of this paper.

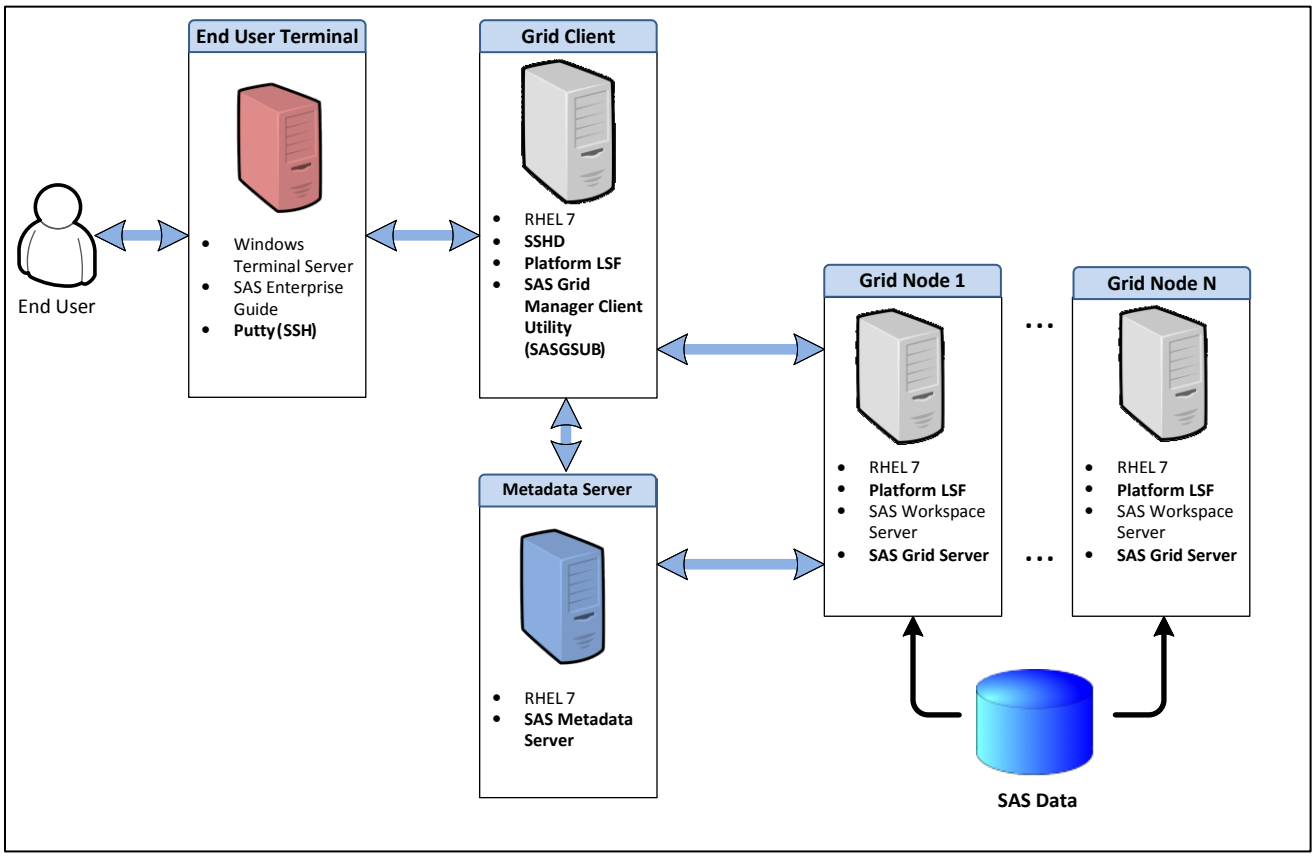


Figure 1. **SAS Grid Manager Client Utility - Application Flow**

ISSUES WITH SASGSUB

Though it is possible to submit a SAS® program in command-line mode using SASGSUB commands, there are a few issues that may need a more customized approach. This especially happens when the user needs to run **a sequence of related SAS® programs** where each program in the sequence requires the previous program to finish (e.g. the next program needs to use the datasets or other products produced in the previous SAS® program). SAS® Grid does not run the submitted group of the programs in the sequence submitted using SASGSUB commands even if all of the programs in the sequence were submitted at the same time in the desired order – manually or by running a system batch file. The reason

is that the SAS® Grid will run each program disregarding the order they were submitted in batch-mode. In fact, each of the programs may be sent to one component of the grid to process. Therefore, while program 1 in a sequence of 3, has not finished the program 2 starts to process. SAS® Grid simply processes each program independently.

One solution to this issue is making a single program using SAS® **%include** statement. The %include statement simply allows to include and run the whole SAS® code from another external file at the location where %include is used in the current SAS® program. When processing, SAS® would see such programs as if the user had copied all of the code in the external file and pasted it into the location in the current program where %include statement was used. The SAS® program that can be submitted using SASGSUB would look like the following:

```
/*-------------------------------------------------------------------
 A SAS program to be submitted to SAS Grid using SASGSUB commands. */
/* This program will include the codes from 3 external programs to be run
  in a sequence:
    1- Program_1.sas
    2- Program_2.sas
    3- Program_3.sas
-------------------------------------------------------------------*/
%include "/my_networkdrive/my_username/my_project1/Program_1.sas";
%include "/my_networkdrive/my_username/my_project1/Program_2.sas";
%include "/my_networkdrive/my_username/my_project1/Program_3.sas";
```

However, one pitfall of this approach is getting a **single log and single output file for all** of the SAS® programs called via %include statement. Most users do prefer to have a separate log and .lst output for each program. A single merged log or merged output is a bit annoying and may make troubleshooting difficult. Traditionally, many users are accustomed to creating a batch file (e.g. in UNIX with a non-GRID SAS® implementation) where they could list the SAS® programs to be submitted in a sequence by simply executing that batch file.

Another issue with this merged log file in a multi-user environment might be getting error messages at the beginning of this merged log file warning the users about the libraries they are not allowed to access. This is particularly the case in multi-user environments where every user/group has access to certain libraries only. These errors are absolutely unrelated to SAS® code submitted. This can happen when a common start-up script is run to define all predefined libraries in a corporate's shared SAS® environment. When using products like SAS® Enterprise Guide®, the user would not usually see such "access denied" errors as these are handled before the user starts to see SAS® EG's graphical interface. In contrast, when using SASGSUB commands, the user would get all of these errors early in the log file.

A third less important issue is that all of the SASGSUB output files are produced in a job directory under the user's username usually created in the directory of the submitted SAS® program. Therefore, the user often needs to transfer these logs and outputs (e.g. .lst file) to original directory of the SAS® program which had been submitted. This is in particular desired in order to keep a SAS® program and its log and output at the same location for future reference. This transfer can be done using either SASGSUB commands or manually.

SUGGESTED SOLUTION

A) THE COMBINATION OF %INCLUDE AND PROC PRINTTO

When submitting a sequence of SAS® programs where the order of submission is also important, one can make a single SAS® program and call each program of interest in the planned order using %include statement. In this scenario, to avoid a single merged log and a merged output (.lst) file from SASGSUB, one can use SAS® **procedure PRINTTO** before and after calling each program and provide new log and output file addresses (pointing to the same directory of each program) to proc PRINTTO. This will re-route the corresponding log and output file to the same directory of the original program. The following SAS® program illustrates this method.

```
options source source2;
/*++++++++++++++++++ Program_Sequence.sas
++++++++++++++++++++++++++++++++++
 A SAS program to be submitted to SAS Grid using SASGSUB commands.
 This program will include the codes from 3 external programs to be run
 in a sequence:
    1- Program_1.sas
    2- Program_2.sas
    3- Program_3.sas
  SAS proc PRINTTO is used to redirect the log and output to the original
  locations of these 3 SAS programs.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
*Change options to allow the code from %include[ed] SAS programs to appear
in the log;
/*--------------------------- Program 1 ------------------------*/
/*Start re-directing the log and output file for Program_1.sas */
proc printto
   log="/my_networkdrive/my_username/my_project1/Program_1.log"
   print="/my_networkdrive/my_username/my_project1/Program_1.lst" new;
run;

%include "/my_networkdrive/my_username/my_project1/Program_1.sas";

/*Close and finish log and output re-direction for Program_1.sas*/
proc printto  log=log print=print;
run;

/*--------------------------- Program 2 ------------------------*/
/*Start re-directing the log and output file for Program_2.sas */
proc printto
   log="/my_networkdrive/my_username/my_project1/Program_2.log"
   print="/my_networkdrive/my_username/my_project1/Program_2.lst" new;
run;

%include "/my_networkdrive/my_username/my_project1/Program_2.sas";

/*Close and finish log and output re-direction for Program_2.sas*/
proc printto  log=log print=print;
run;

/*--------------------------- Program 3 ------------------------*/
/*Start re-directing the log and output file for Program_3.sas */
proc printto
```

```
   log="/my_networkdrive/my_username/my_project1/Program_3.log"
   print="/my_networkdrive/my_username/my_project1/Program_3.lst" new;
run;

%include "/my_networkdrive/my_username/my_project1/Program_3.sas";

/*Close and finish log and output re-direction for Program_3.sas*/
proc printto  log=log print=print;
run;
```

The SAS® program above (Program_Sequence.sas) is saved on the disk. Here for reference purposes, this (Program_Sequence.sas) is called **the mother or amalgamated SAS® program**, and the included programs are referred to as **child SAS® programs** (e.g. Program_1.sas to Program_3.sas in example above). The mother SAS® program can be submitted to SASGSUB in a shell terminal (e.g. PuTTY) for example:

```
SASGSUB -GRIDSUBMITPGM /my_networkdrive/my_username/my_project1/Program_Sequence.sas
```

The mother SAS® program does not have to lie in the same directory of the original programs in the sequence. It is assumed that SASGRID has been granted the permissions to access and write the resulting log and output files at the user's private space (e.g. /my_networkdrive/my_username/my_project1). In the example above, SASGSUB still creates a job directory (e.g. in the original location of the mother/amalgamated SAS® program). In this directory, a merged log file can still be found. However, this merged file is now much shorter and has information about what happened at SAS® job start-up and SAS® log and output redirections conducted by proc PRINTTO.

Use of "**options source source2**" at the first line of the mother SAS® program helps the SAS® program codes appended using %include statement would appear in the final log files.

Advantages of this approach:

1. The SAS® programs listed using %include statement will run in the listed "order" one by one.
2. Separate log and output files for each child SAS® program is saved right next to each of the child programs. This also saves the user's time to copy the log and output from SASGSUB's job directory back to the original location of each of the submitted SAS® programs.
3. The errors relating to SAS® System start-up (e.g. errors due to automatic definition of libraries the user may not be permitted to access) will appear in a higher level log file in the job directory of SASGSUB. This will prevent the user seeing any of these higher level errors unrelated to his/her SAS® code.

Important notes and limitations of this approach:

1. It is recommended that before submitting the mother/amalgamated SAS® program (e.g. Program_Sequence.sas in example above) to SASGSUB, the user ensures that all of the child SAS® programs and the mother SAS® program are not open in any other application (e.g. any text editor, SAS® Enterprise Guide®, etc.). SASGSUB may not like the **file locks** set on these program files when accessing them. This issue, however, may vary in different operating systems.

2. SAS® would see all of the code from child SAS® programs amalgamated using %include statement into the mother SAS® program as a single program. The user needs to make sure the dataset names (especially those created in WORK directory), general macro variables, and other SAS® objects in one program would not conflict with those in previous or next programs.
3. A very important point to remember is **avoiding use of proc PRINTTO within the child SAS® program**. This can interrupt the log re-direction already planned within the mother program and will cause fragmented and confusing log streams.
4. The SAS® job which runs the code synthesized in the mother program uses a single temporary WORK directory. Therefore, if big enough, the piling up temporary datasets produced in this process may fill up the temporary space in WORK directory. This issue may be solved by deleting all datasets in WORK directory after the code for each child program in the sequence has finished running, and before the code of the next child program starts to run - assuming that the user is not using the WORK directory to save any datasets to be used by the next program in a sequence. This **inter-program cleanup** operation is possible via use of **proc DATASETS**.

B) AUTOMATING SAS® BATCH PROGRAM GENERATION USING AN IN-HOUSE TOOL

Although the suggested solution above solves the issues some users may experience with SASGSUB, generation of the code for the mother SAS® program and synthesizing the codes from all child programs in a sequence using proc PRINTTO (and proc DATASETS) may seem time-consuming and boring. To facilitate this process an in-house tool was developed introduced below.

This browser-based tool called "**SAS® GSUB Batch Code Maker"** was developed using HTML, JavaScript and CSS programming. It is a lightweight tool implemented in a single html file, and can be run as a web page in major browsers with JavaScript capability (i.e. Internet Explorer, Firefox and Chrome). It can be run locally without the need for a web-server. In our environment, this tool (i.e. HTML file) was put in a file directory accessible in server list of SAS® Enterprise Guide®, so that the users can even open and use it within SAS® Enterprise Guide's browser.

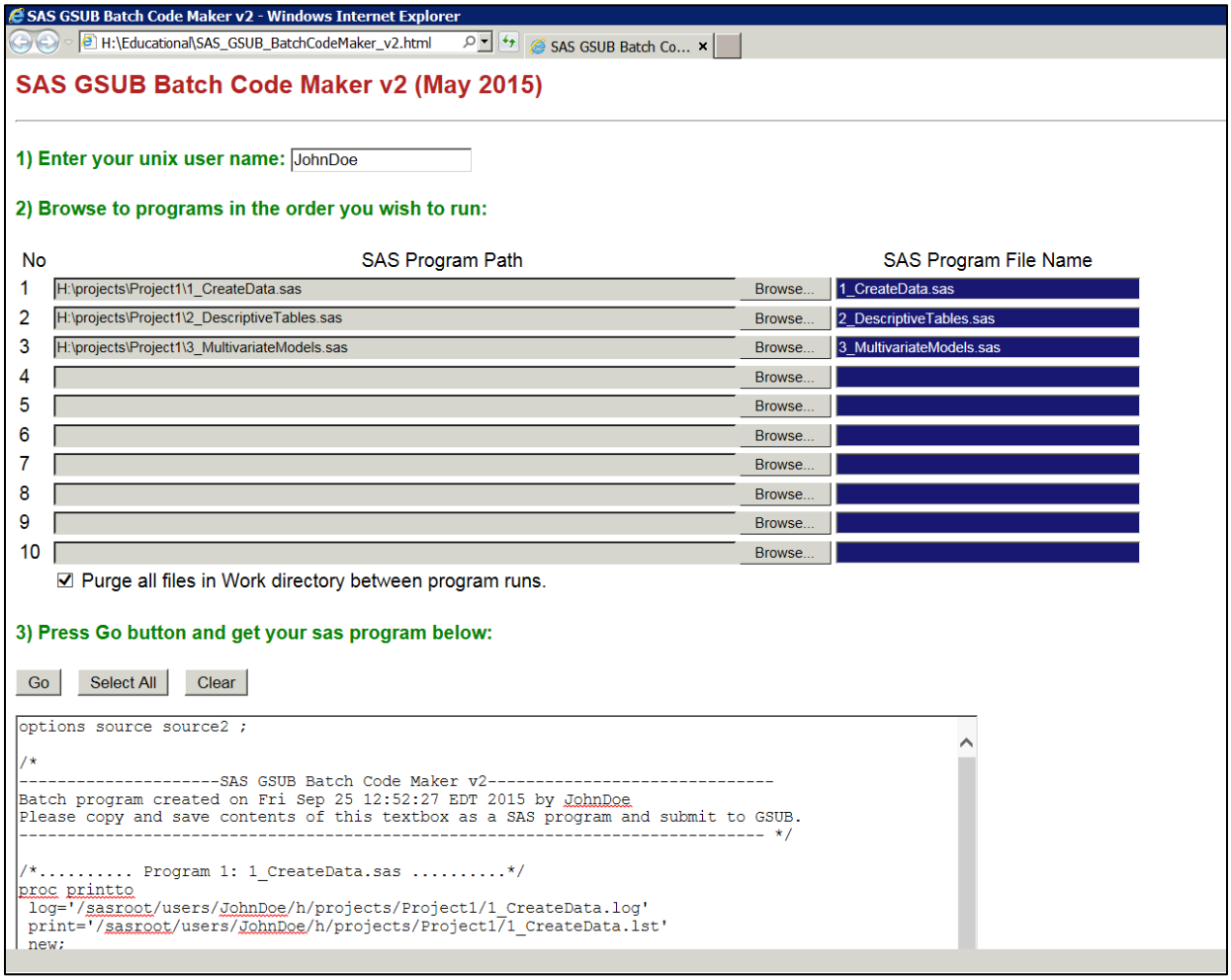
To function, the tool only asks the user to locate the child SAS® programs in the desired sequence of running. It looks like attaching files one by one when sending an email. The user then presses a Go button and gets all of the amalgamated SAS® code for the mother SAS® program (e.g. Program_Sequence.sas in example above) in a textbox (Display 1). This code can be copied back to a code editor (e.g. any text editor), saved as a new SAS® program on disk, and finally be submitted to SASGSUB.

The tool translates the local program paths on user's local machine/account (e.g. *"P:/myproject_1/Program_1.sas"*) to the network paths understandable (resolvable) to SAS® Grid and SASGSUB (e.g. "*/corporate_sasroot/users/my_user_name/myproject_1/Program_1.sas*"). For this purpose, the tool asks the user name to make the path to the user's space on a network drive. The JavaScript code within the tool can be altered using any text/code editor to change these path translations depending on local settings of a shared environment different from ours.

For the tool to work properly, the security settings of the browser should allow the browser to show the "full path" to the child programs (and not just file names) in the file objects on the user interface. This is usually the case by default. However, if a browser's security settings are very strict, it may not show the full path by default. If so, the browser's security settings may need changing by the user. This is usually possible from within the browser. For instance in Internet Explorer®, this can be achieved by: Tools →

Internet Options → Security → Custom level → Enable “Include local directory path when uploading files to a server”.

A check box also gives the user the option to ask the too to produce aforementioned inter-program data cleaning operation using proc DATASETS. If checked, the code will include a proc DATASETS block which will delete all datasets in the WORK directory to release space.



Display 1. A screenshot of HTML/JavaScript tool “*SAS® GSUB Batch Code Maker*” used to automate SAS® batch program code generation

CONCLUSION

Submission of a sequence of SAS® programs in batch mode to SAS® Grid (SASGSUB) and getting back individual log and output files is possible using built-in SAS® tools and capabilities. Creative use of a client-side browser-based program can facilitate code generation for this purpose and save programmers’ time.

REFERENCES

SAS Institute Inc. 2014. **Grid Computing in SAS® 9.4, Third Edition.** Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Mohammad-Reza Rezai
Institute for Clinical Evaluative Sciences
G1 06, 2075 Bayview Avenue
Toronto, Ontario M4N 3M5
Canada
reza.rezai [at] ices.on.ca
https://www.linkedin.com/in/mrezarezai/