PhUSE 2017

Paper CT14

Job of Jobs

Kris Lauwers, Johnson and Johnson, Beerse, Belgium
Raphael Noirfalise, Johnson and Johnson, Beerse, Belgium

ABSTRACT

The ability to run SAS® programs simultaneously, in batches, is a major efficiency-gaining feature, widely used within clinical trial analysis. For local- and server-based SAS environments, several options are available to achieve this. Facing a migration towards SAS cloud-based environments, SAS Drug Development (SDD) and later SAS Life Science Analytics Framework (LSAF), new options need to be explored for batch submitting SAS programs. In this paper, we would like to present a novel and simple solution to programmatically batch submit multiple SAS programs, either simultaneously or consecutively, and discuss the applications for which this solution can be deployed.

INTRODUCTION

In a local- or server-based SAS environment, several options exist that enable the user to batch submit multiple SAS programs. Often this is a crucial timesaving feature when facing a major deadline. When moving towards the cloud-based environment SAS provides, no out-of-the-box efficient solution exists to run programs in a batch. Here, we present a new way to programmatically run and schedule several SAS programs in one go.

LSAF has some fundamental differences with the familiar local- and server-based SAS environments regarding batch submission of jobs. To that end, the first step in LSAF is creating a job that runs your SAS program. Within one job you can specify multiple SAS programs to be run consecutively and define global macro parameters to be used in your programs. However, LSAF will publish the output only when the job is complete. Hence, we cannot run a SAS program that creates a dataset, and then, within the same job, run another program that uses this dataset. Several new, user-friendly applications (LSAF Macro Application Programming Interface or API, see references) are developed by SAS to facilitate job-of-jobs.

JOB OF JOBS

With these features of LSAF in mind, we developed a job-of-job method, where we have one 'parent' job that runs a 'parent' SAS program. Within that 'parent' SAS program, 'child' SAS jobs are submitted that contain the 'child' SAS programs we want to run. As we want to avoid copy-pasting of job names, we automatically read the job files located in a prespecified folder and put the file names in a dataset. When applying consistent job names, we can use wildcards to select the jobs and hence the programs we want to run.

USE IN CLINICAL TRIAL STATISTICAL PROGRAMMING SETTING

We prepared a parent SAS job and a program to enable the user to easily run and schedule in one go all the programs that create CDISC ADaM datasets and all tables/listings/figures (TLF) needed for the Clinical Study Report (CSR). For each ADaM dataset and TLF a SAS job and program exists with the prefix AD, T, L and F. When running the 'parent' job, the only action the user needs to perform is to indicate the prefix of the jobs he wants to run. In the J&J programming environment, all the jobs are in the same study-specific job folder. Through a getchildren API, all the jobs in that folder are listed in a SAS dataset and by applying the wildcards provided in the job, the list is reduced to the jobs the user wants to run. Then, the number of remaining jobs in the list is counted and that count x is used to loop around a submitjob API that calls and runs the x^{th} job.

While running each job, the status of the job provided by LSAF is collected in a dynamic macro parameter. This allows SAS to be put to sleep until the job is finished successfully. This is needed for the TLF programs to be able to use the ADAM datasets, which are only published at the end of each separate ADaM job. Without putting SAS to sleep, all the jobs and programs will start running quasi simultaneously.

USE FOR RUNNING LARGE PROGRAMS / SIMULATIONS

As it is the case in every SAS environment, running heavy programs on large datasets or running simulations can take a very long time. In our experience, it is not uncommon to have to wait several days for a program or simulation to complete. The solution given above allowed us to drastically reduce the runtime of such analysis from several days to several hours, by cutting the analysis or simulation in pieces and running these pieces simultaneously. What happens is that each job will be run as if it came from a different user and each job will be sent to one of the servers SAS employs depending on the load of these servers.

CONCLUSION

The job-of-jobs programming solution is an easy to use tool that enables the users of cloud-based SAS environment to run multiple programs in batches. Furthermore, the principle behind this tool opens opportunities for running large scale analysis.

REFERENCES

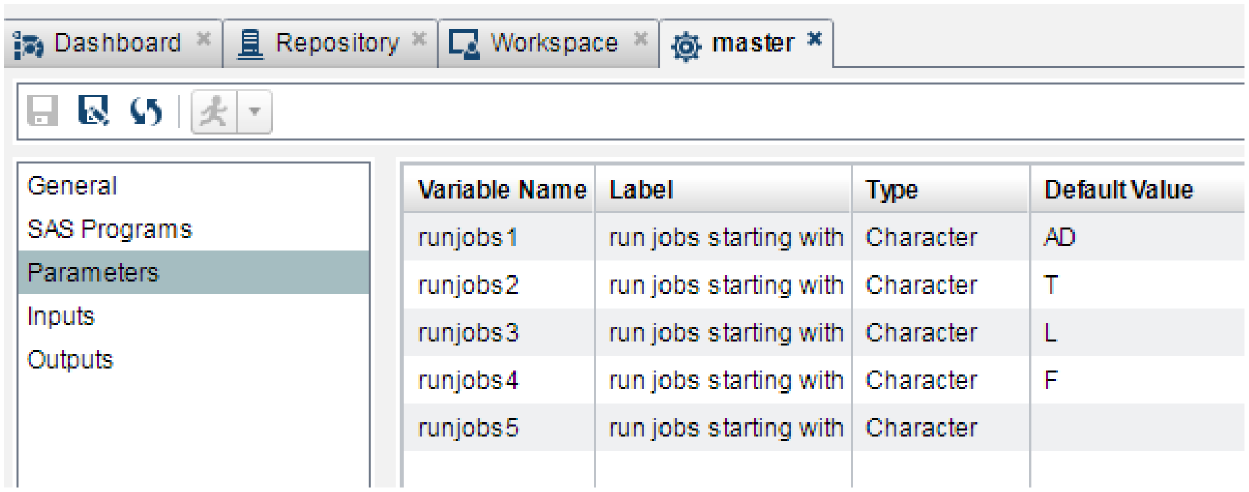
LSAF MACRO APPLICATION PROGRAMMING INTERFACE (API)

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Kris Lauwers
Johnson & Johnson
Turnhoutseweg 30
2340 Beerse Antwerp (Belgium)
Email: klauwer2@its.jnj.com

APPENDIX

LSAF JOB AND SAS PROGRAM CODE



Variable Name	Label	Type	Default Value
runjobs1	run jobs starting with	Character	AD
runjobs2	run jobs starting with	Character	T
runjobs3	run jobs starting with	Character	L
runjobs4	run jobs starting with	Character	F
runjobs5	run jobs starting with	Character	

```
*************************************************************************;
*** master.sas (SAS program indicated in the master.job above)***;
*************************************************************************;
%let joblocation = %str (/Users/KLAUWER2/jobs);
*** get macro parameters from master.job;
%lsaf_getjobparameters(
  lsaf_path=%cmpres(&joblocation/master.job),
  sas_dsname=master_jobparameters);
data work.jobparameters;
  set master_jobparameters;
  rename defaultvalue=value;
run;
*** get all jobs related to each seperate program you want to run;
%lsaf_getchildren(
  lsaf_path=%cmpres(&joblocation),
  lsaf_recursive=1,
  lsaf_dsname=jobs1);
*** select the jobs you want to run;
data jobs2;
  set jobs1;
  if name in:("%cmpres(&runjobs1)") or
     name in:("%cmpres(&runjobs2)") or
     name in:("%cmpres(&runjobs3)") or
     name in:("%cmpres(&runjobs4)") or
     name in:("%cmpres(&runjobs5)");
  if index(upcase(objecttype),"JOB")>0;
run;
*** count the number of jobs selected;
proc sql noprint;
  select count(*) into:njobs
  from jobs2;
quit;
*** run every job consecutively;
%macro loop;
  %do x=1 %to &njobs;
    data _null_;
      set jobs2;
      if _n_=&x then call symput("job&x",trim(left(name));
    run;
    %lsaf_submitjob(
      lsaf_path=%cmpres(&joblocation/&&job&x),
      sas_dsname=work.jobparameters);
    *** put SAS to sleep until submission status of &&job&x is set to
        "completed";
    *** omitting this step will let you run all selected jobs simultaneously;
    %tryagain:
    *** let SAS sleep for 6 seconds;
    data _null_;
      call sleep (6,1);
    run;
    *** check the submission status;
    %let id&x=&_lsafJobSubmissionId_;
    %lsaf_getsubmissionstatus( lsaf_jobsubmission_id=&&id&x);

    %if (%index(%upcase (&_lsafJobSubmissionStatus_),%STR(COMPLETED))<=0)
      %then %goto tryagain;
  %end;
%mend loop;
%loop;
```