Paper SM11

%batchrunprograms: SAS Studio Background Submit on Multiple Programs

Luis Andre Soriano, OCS Consulting B.V., 's-Hertogenbosch, The Netherlands

ABSTRACT

As with traditional local SAS® Batch Submit, SAS Studio Background Submit on a program is important in ensuring the program runs successfully from a clean state and contains no errors, warnings, or undesirable notes. Often, multiple programs need to be run sequentially, independently, and frequently (e.g. developing SDTMs/ADaMs/TLFs). There is unfortunately no built-in way in SAS Studio to Background Submit multiple programs. Current solutions involve tinkering with XCMD option (security risk), creating non-SAS tools (separate skill to learn), or using %include on multiple programs (independence hard to ensure).

In this presentation we'll show that **%batchrunprograms** is developed by only incorporating tools from SAS/CONNECT® and Base SAS (e.g. ODS, PROC PRINTTO). Users can set up maintained lists of programs that can be run sequentially and independently any time, with functionalities of stopping the batch run in case of errors, skipping some programs, and providing users summary of the batch run.

INTRODUCTION

It is important to run a program from start to finish in a fresh and clean session (i.e. without temporary macro variables, datasets, options, etc. active) so that the result of the program is consistent and reproducible, and actual errors, warnings, or undesirable notes are captured. To ensure that a run session is clean, some simple ways include a “Batch Submit” (with local Base SAS) or a “Background Submit” (with SAS Studio). This is quick and easy to do one-by-one with few programs but gets increasingly tedious to do with multiple programs. In most projects, there will be many times where you will need to run multiple programs sequentially, independently, and frequently (e.g. clinical trial deliverables over the project’s lifespan).

For SAS Studio users who run their programs on servers, there is currently no built-in way to do Background Submit on multiple programs sequentially and independently. Some solutions have been proposed such as enabling XCMD in default options (which is a security risk on servers), creating custom tools outside SAS (which requires a separate skill and budget to create), or using %include on multiple programs (which always has a risk of one program affecting another program unintentionally, aside from not being able to save the log and result of each program).

%batchrunprograms is a tool that satisfies all the requirements of a good sequential and independent run—there is a maintained list of programs to batch run; every program runs on a fresh and clean session without affecting the other programs; and the log and result files of each program are saved. There are also added functionalities for producing a summary of the entire run showing success, errors, and warnings; stopping the batch run in case of errors on critical programs; and skipping some programs. These are done only with the basic functionalities of Base SAS, SAS/CONNECT, and Microsoft Excel®.

PREPARATION

The user lists all the programs to batch run in the accompanying Excel file of the tool. The user supplies the full path, program name, whether they want to stop the batch run if some programs fail, and whether they want to skip some programs. Relative paths work, as long as the program invoking the **%batchrunprograms** configures the relative pathing accordingly. Forward slash (/) and backward slash (\) are both accepted. Extension on program names is optional. The user can save multiple of these Excel files for different purposes (e.g. one for SDTM programs, another one for ADaM programs, etc.). The tool will run the programs one by one from top to bottom.

PATH	PROGNAME	FAILSTOP	DONOTRUN
.	create_combined_input_sdtm	X	
P:\Projects\Client\Study ID\Development\Programs	create_dummied_input_sdtm	X	
P:\Projects/Client\Study ID/Development\Programs	tdomains_to_sas	X	
P:/Projects/Client/Study ID/Development/Programs	create_sdtm_dm	X	
.	create_sdtm_se.sas	X	
.	create_sdtm_sv.sas	X	
.	create_sdtm_ae		
.	create_sdtm_ag		
.	create_sdtm_cm		
.	create_sdtm_cp		
.	create_sdtm_ds		
.	create_sdtm_dv		X
.	create_sdtm_ec		X

Once the Excel files are set up, the user invokes the **%batchrunprograms** to point to the Excel files. An example is shown below:

```
%batchrunprograms(proglist=./_sdtm.xlsx);
%batchrunprograms(proglist=P:/Projects/Client/Study ID/Development/Programs/_adam.xlsx);
```

THE SOLUTION

To achieve complete independence of the program runs, **%batchrunprograms** uses SAS/CONNECT functionalities. The tool runs each program in their own separate remote session. A remote session is a session outside the local session (the one you are running the **%batchrunprograms** from). It's a clean session that is not affected by the local session, and can be closed once a program has been run on it.

The tool also uses basic SAS functionalities to loop through the list of programs and execute them, save the logs and results into external files, and capture the errors/warnings that each program might have. The tool then reports whether each program run is successful or not.

For each program, the following step-by-step process is executed.

1. Open a clean remote session and sign on to it. This session will not inherit the temporary objects from the local session. It does not need a true remote server (with user ID and password) as it is only a separate SAS process on the same machine of the local session.

   ```
   signon remote0 sascmd="!sascmd -nosyntaxcheck -noterminal" cmacvar=remote0_chk;
   ```

2. Supply the necessary information from the local session to the remote session. This includes the SAS program name, log name, output name, and error and warning flags. This process shows how the local session can communicate with the remote session in a controlled manner.

   ```
   %syslput file     = &&file&i.;
   %syslput log      = &&log&i.;
   %syslput output   = &&output&i.;
   %syslput berr     = &berr.;
   %syslput bwarn    = &bwarn.;
   ```

3. Begin the submission of codes in the remote session. Set the local session to wait for the remote session to finish before continuing. All codes after this line will then be submitted to the separate remote session and not to the local session.

   ```
   rsubmit remote0 wait=yes;
   ```

4. Configure the ODS to store the results in HTML and save it as an external file. The tool saves the result page into an HTML file to mimic the default behavior of a normal SAS Studio Background Submit. The user can choose to save in different file format like PDF if preferred. The GPATH and ODS GRAPHICS are important to check because of the behavior of output processing in remote sessions. This is particularly important for programs that output plots.

   ```
   ods listing close;
   ods html file = "&output." gpath = "%sysfunc(getoption(work))";
   ods graphics on;
   ```

5. Redirect the remote session log into an external log file. This is done to capture the log of the remote session run, as it is by default displayed on the local session log.

```
proc printto log="&log." new; run;
```

6. Execute the entire program in the clean remote session.

```
%include "&file.";
```

7. Capture the last error text and the last warning text from the remote session and store it in the local session. This is so we can report the status of each program run. This process shows how the remote session can communicate with the local session in a controlled manner.

```
%nrstr(%sysrput fileerr =&&sys&berr.text);
%nrstr(%sysrput filewarn=&&sys&bwarn.text);
```

8. End the submission of codes in the remote session. Set the local session to wait for the remote session to finish before continuing. All codes after this line will then be submitted to the local session and not to the separate remote session.

```
endrsubmit;
waitfor _all_;
```

9. Sign off from the remote session and close it.

```
signoff _all_;
```

EXTRA FUNCTIONALITIES

On top of providing the batch run summary, the tool has functionalities that help the user control the behavior of the batch run. As an example, take the list of programs shown below:

1	PATH	PROGNAME	FAILSTOP	DONOTRUN
2	.	study0001_dm.sas	X	
3	.	study0001_se		X
4	.	study0001_sv		
5	.	study0001_ex		
6	.	study0001_fa		
7	.	study0001_ae	X	
8	.	study0001_tplot1		

SUMMARY OF THE BATCH RUN

Once the user runs the program to invoke **%batchrunprograms**, the results page (if done in an interactive run) or the HTML results file (if done in SAS Studio Background Submit) will show status of each program, whether it ran successfully or with errors/warnings.

Results of Batch Run for:
./study0001_sdtm.xlsx
Run: 2025-09-03T12:09:00

Obs	Excel Row	PATH	PROGNAME	RESULTS
1	2	.	study0001_dm	Has warning(s).
2	4	.	study0001_sv	Has error(s).
3	5	.	study0001_ex	Successful Run.
4	7	.	study0001_ae	Successful Run.
5	8	.	study0001_tplot1	Successful Run.

STOPPING THE BATCH RUN

Setting FAILSTOP="X" on programs will stop the batch run when those programs do not exist or have error(s). For example, study0001_ae is created to have errors. This results to **%batchrunprograms** not running study0001_tplot.sas as shown in the summary of the batch run:

Results of Batch Run for:
./study0001_sdtm.xlsx
Run: 2025-09-03T11:46:41

Obs	Excel Row	PATH	PROGNAME	RESULTS
1	2	.	study0001_dm	Has warning(s).
2	4	.	study0001_sv	Has error(s).
3	5	.	study0001_ex	Successful Run.
4	6	.	study0001_ae	Has error(s).

```
ERROR: [redacted]/Development/Programs/study0001_ae.sas
ERROR: is a critical file and has an error.
ERROR: Utility will STOP now.
```

If a program does not exist but is not set as FAILSTOP="X", then the tool will report it but will continue to run the next programs. For example, study0001_fa.sas does not exist but is not set to FAILSTOP="X".

```
WARNING: [redacted]/Development/Programs/study0001_fa.sas
WARNING: is not a critical file (FAILSTOP is not X) and does not exist.
WARNING: Utility will not run this program but will proceed to next programs.
WARNING: ( Excel row = 6 )
```

SKIPPING PROGRAMS

Setting DONOTRUN="X" on programs will skip those programs. For example, study0001_se is set to have DONOTRUN="X", and the summary of the batch run shows that it is skipped:

Results of Batch Run for:
./study0001_sdtm.xlsx
Run: 2025-09-03T12:09:00

Obs	Excel Row	PATH	PROGNAME	RESULTS
1	2	.	study0001_dm	Has warning(s).
2	4	.	study0001_sv	Has error(s).
3	5	.	study0001_ex	Successful Run.
4	7	.	study0001_ae	Successful Run.
5	8	.	study0001_tplot1	Successful Run.

CONCLUSION

%batchrunprograms is a tool that enables the user to create list(s) of programs and execute the programs sequentially and independently whenever desired. It runs each program in a completely separate and clean remote sessions, saves the log and result as files, and provides a summary of the batch run. It has functionalities to skip programs and to stop the batch run when a critical program does not exist or has errors.

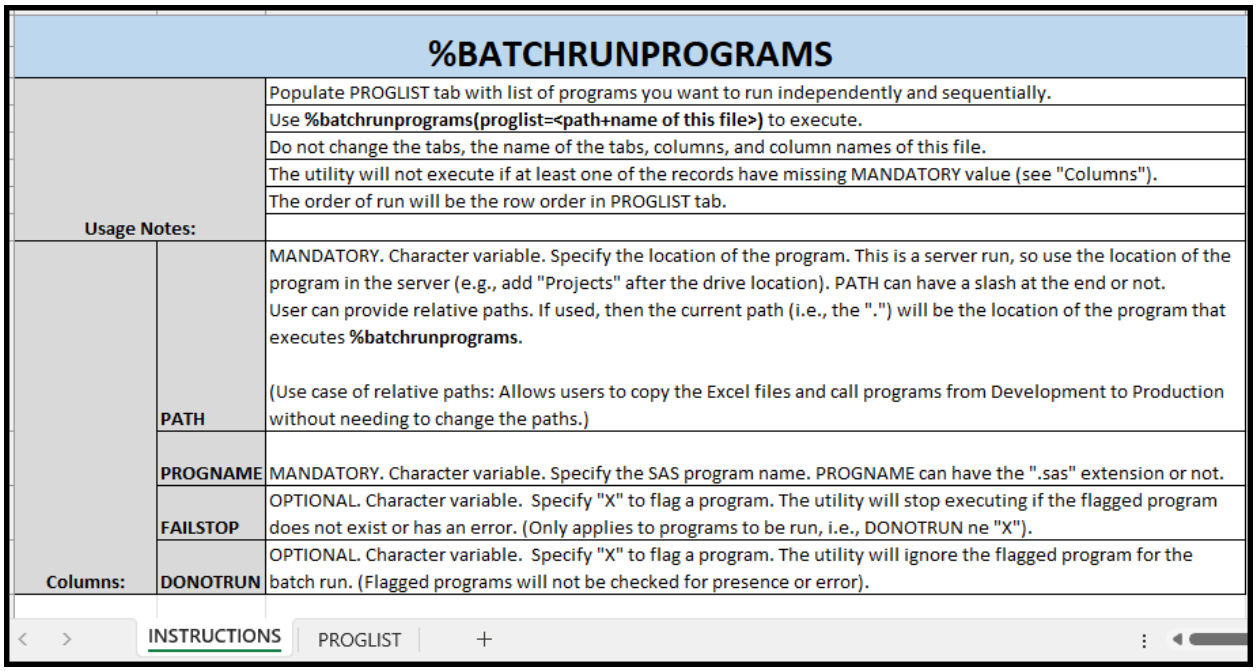
APPENDIX I (%BATCHRUNPROGRAMS)

```
%macro batchrunprograms(proglist=);

  options noquotelenmax missing="" validvarname=upcase;
  %let datetime = %sysfunc(datetime(),is8601dt.);
  %let proglist = %str(&proglist.);
  %let mexit    = 0;
  %let berr     = %sysfunc(compress(er ror));
  %let bwarn    = %sysfunc(compress(war ning));

  /* Mandatory parameters */
  %if %length(&proglist.) = 0 %then %do;
    %put %str(ERR)%str(OR:) Parameter PROGLIST is empty. Please provide the Excel file containing
list of programs.;
    %let mexit = 1;
    %goto absexit;
  %end;
  %if %sysfunc(fileexist(&proglist.)) = 0 %then %do;
    %put %str(ERR)%str(OR:) The Excel file supplied in PROGLIST does not exist. Please double
check.;
    %let mexit = 1;
    %goto absexit;
  %end;

  /* Create dataset to hold the results of run */
  data _run_results (where = (not missing(path)));
    array __temp2 {*} 8. row;
    array __temp {*} $200. path progname results;
  run;

  libname bfile xlsx "&proglist.";

  data _rs0 (where = (cmiss(path, progname) = 0 and strip(upcase(donotrun)) ne "X")
             keep = order row path progname fileref rc absolutepath fullfilepath fulllogpath
fulloutputpath failstop is_exist donotrun);
    length fileref absolutepath fullfilepath fulllogpath fulloutputpath $200.;
    set bfile.proglist;
    order = _n_;
    row   = _n_+1;
    failstop = strip(upcase(failstop));
    donotrun = strip(upcase(donotrun));
    if failstop ne "X" then failstop = "Z";
    if donotrun ne "X" then donotrun = "Z";
    /* Remove slash at the end of PATH if there is any */
    path = strip(path);
    if strip(substr(reverse(path),1,1)) in ("/" "\") then path =
strip(substr(path,1,length(path)-1));

    /* Remove extention from PROGNAME */
    progname = strip(progname);
    progname = strip(scan(progname,1,"."));

    /* Err.or checks */
    if missing(path) and not missing(progname) then do;
      put "ERR" "OR: PATH is missing from Excel row = " row "of the program list. Please
populate! Utility will NOT proceed.";
      call symputx("mexit", 1);
    end;

    if not missing(path) and missing(progname) then do;
      put "ERR" "OR: PROGNAME is missing from Excel row = " row "of the program list. Please
populate! Utility will NOT proceed.";
      call symputx("mexit", 1);
    end;

    if not missing(path) and not missing(progname) then do;
      fileref         = "r"||strip(put(row, best.));
      rc              = filename(strip(fileref), strip(strip(path)));
      absolutepath    = strip(tranwrd(tranwrd(pathname(fileref),"\","/"),"/.sas",""));
      fullfilepath    = strip(absolutepath)||'/'||strip(progname)||".sas";
      fulllogpath     = strip(absolutepath)||'/'||strip(progname)||".log";
      fulloutputpath  = strip(absolutepath)||'/'||strip(progname)||".html";
    end;
    if not missing(fullfilepath) then is_exist = fileexist(fullfilepath);
```

```
  run;

  %if &mexit. = 1 %then %goto exit;

  data _rs1 (where = (is_exist = 1 and until_here = 0));
    retain until_here 0;
    set _rs0;
    by order;

    /*
    If a program does not exist but not a critical file (FAILSTOP is not X), then alert user but
proceed with other files.
    If it is a critical file (FAILSTOP=X), then alert user and only run right before that
program.
    */

    if is_exist = 0 and strip(upcase(failstop)) = "X" then do;
      until_here = 1;
      put "WAR" "NING: ------------------------------------------------------------------------";
      put "WAR" "NING: " fullfilepath;
      put "WAR" "NING: is a critical file (FAILSTOP=X) and does not exist.";
      put "WAR" "NING: Utility will run the programs UNTIL before this file.";
      put "WAR" "NING: ( Excel row = " row ")";
      put "WAR" "NING: ------------------------------------------------------------------------";
    end;

    if is_exist = 0 and strip(upcase(failstop)) ne "X" and until_here = 0 then do;
      put "WAR" "NING: ------------------------------------------------------------------------";
      put "WAR" "NING: " fullfilepath;
      put "WAR" "NING: is not a critical file (FAILSTOP is not X) and does not exist.";
      put "WAR" "NING: Utility will not run this program but will proceed to next programs.";
      put "WAR" "NING: ( Excel row = " row ")";
      put "WAR" "NING: ------------------------------------------------------------------------";
    end;
  run;

  proc sql noprint;
    select fullfilepath   into :files     separated by "|" from _rs1;
    select fulllogpath    into :logs      separated by "|" from _rs1;
    select fulloutputpath into :outputs   separated by "|" from _rs1;
    select failstop       into :failstops separated by "|" from _rs1;
    select path           into :paths     separated by "|" from _rs1;
    select progname       into :prognames separated by "|" from _rs1;
    select row            into :rows      separated by "|" from _rs1;
  quit;

  %let countfiles = %sysfunc(countw(&files,|));

  %do i = 1 %to &countfiles;
    %let file&i.      = %scan(&files,&i.,|);
    %let log&i.       = %scan(&logs,&i.,|);
    %let output&i.    = %scan(&outputs,&i.,|);
    %let failstop&i.  = %scan(&failstops,&i.,|);
    %let path&i.      = %scan(&paths,&i.,|);
    %let progname&i.  = %scan(&prognames,&i.,|);
    %let row&i.       = %scan(&rows,&i.,|);

    signon remote0 sascmd="!sascmd -nosyntaxcheck -noterminal" cmacvar=remote0_chk;
    %syslput file     = &&file&i.;
    %syslput log      = &&log&i.;
    %syslput output   = &&output&i.;
    %syslput berr     = &berr.;
    %syslput bwarn    = &bwarn.;
    rsubmit remote0 wait=yes;
    ods listing close;
    ods html file = "&output." gpath = "%sysfunc(getoption(work))";
    ods graphics on;
    proc printto log="&log." new; run;
    %include "&file.";
    %nrstr(%sysrput fileerr =&&sys&berr.text);
    %nrstr(%sysrput filewarn=&&sys&bwarn.text);
    endrsubmit;
    waitfor _all_;
    signoff _all_;

    data _run_results0;
```

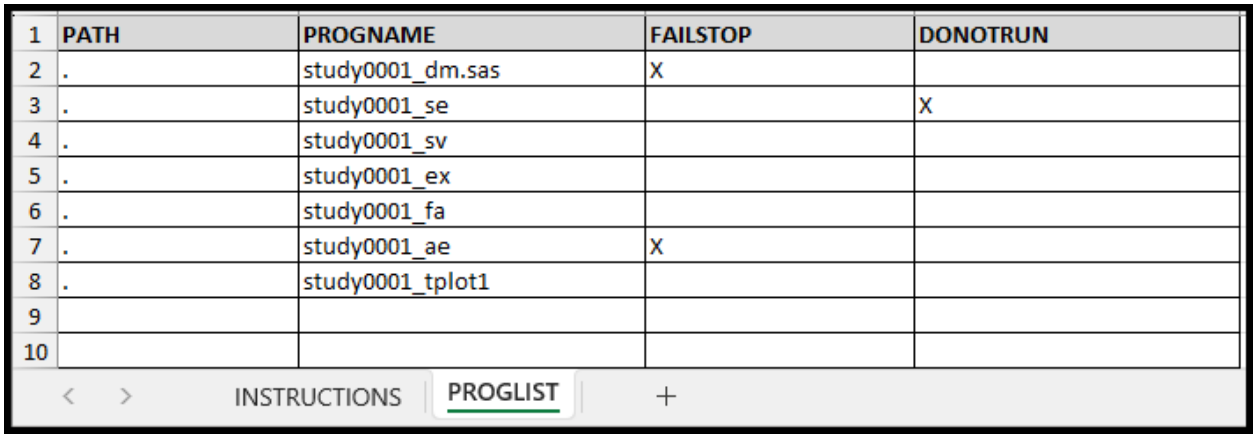
```
      length path progname results $200.;
      row         = &&row&i.;
      path        = "&&path&i.";
      progname    = "&&progname&i.";
      results     = "";
      %if %length(&fileerr.)=0 and %length(&filewarn.)=0 %then %do;
        results = "Successful Run.";
      %end;
      %if %length(&fileerr.)>0 %then %do;
        results = strip(strip(results)||" Has err"||"or(s).");
      %end;
      %if %length(&filewarn.)>0 %then %do;
        results = strip(strip(results)||" Has war"||"ning(s).");
      %end;
    run;

    proc append base = _run_results data = _run_results0 force; run;

    %if %length(&fileerr.)>0 and %upcase(&&failstop&i.) = X %then %do;
      %put %str(ERR)%str(OR: &&file&i.);
      %put %str(ERR)%str(OR: is a critical file and has an err)%str(or.);
      %put %str(ERR)%str(OR: Utility will STOP now.);
      %goto exit;
    %end;

  %end;

  %exit:
  title "Results of Batch Run for:";
  title2 "&proglist.";
  title3 "Run: &datetime";
  proc print data = _run_results label;
    label row         = "Excel Row"
          path        = "PATH"
          progname    = "PROGNAME"
          results     = "RESULTS"
          ;
  run;
  title;
  title2;
  title3;

  libname bfile clear;
  proc datasets noprint kill lib = work memtype = data; run;

  %absexit:

%mend batchrunprograms;
```

APPENDIX II (EXAMPLE OF THE EXCEL FILE OF LIST OF PROGRAMS)



%BATCHRUNPROGRAMS		
Usage Notes:		Populate PROGLIST tab with list of programs you want to run independently and sequentially. Use **%batchrunprograms(proglist=<path+name of this file>)** to execute. Do not change the tabs, the name of the tabs, columns, and column names of this file. The utility will not execute if at least one of the records have missing MANDATORY value (see "Columns"). The order of run will be the row order in PROGLIST tab.
Columns:	**PATH**	MANDATORY. Character variable. Specify the location of the program. This is a server run, so use the location of the program in the server (e.g., add "Projects" after the drive location). PATH can have a slash at the end or not. User can provide relative paths. If used, then the current path (i.e., the ".") will be the location of the program that executes **%batchrunprograms**. (Use case of relative paths: Allows users to copy the Excel files and call programs from Development to Production without needing to change the paths.)
	PROGNAME	MANDATORY. Character variable. Specify the SAS program name. PROGNAME can have the ".sas" extension or not.
	FAILSTOP	OPTIONAL. Character variable. Specify "X" to flag a program. The utility will stop executing if the flagged program does not exist or has an error. (Only applies to programs to be run, i.e., DONOTRUN ne "X").
	DONOTRUN	OPTIONAL. Character variable. Specify "X" to flag a program. The utility will ignore the flagged program for the batch run. (Flagged programs will not be checked for presence or error).

INSTRUCTIONS PROGLIST +



	PATH	PROGNAME	FAILSTOP	DONOTRUN
2	.	study0001_dm.sas	X	
3	.	study0001_se		X
4	.	study0001_sv		
5	.	study0001_ex		
6	.	study0001_fa		
7	.	study0001_ae	X	
8	.	study0001_tplot1		
9				
10				

INSTRUCTIONS PROGLIST +

CONTACT INFORMATION

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

Author Name: Luis Andre Soriano
Company: OCS Consulting B.V.
Address: Ruwekampweg 2 g, 5222 AT, 's-Hertogenbosch, The Netherlands
Work Phone: +31 (0)73 523 6000
Email: luisandre.soriano@ocs-consulting.com
Website: www.ocs-lifesciences.com