PharmaSUG 2018 - Paper AD-26

Making the most of SAS® Jobs in LSAF®

Sonali Garg, Alexion; Greg Weber, DataCeutics

ABSTRACT

SAS Life Science Analytics Framework (LSAF) provides the ability to have a 21 CFR Part 11 compliant web-based Statistical Computing Environment (SCE) where studies data can be stored securely with versioning and audit history capabilities. Jobs are a critical part of an LSAF environment as programs must be run via jobs. Starting in version 4.7.1, the Run and Populate feature is a new and improved way of creating and running jobs. This paper will present our experiences with creating jobs prior to 4.7.1, the issues we encountered and how Run and Populate resolves these concerns and helps us write efficient jobs reducing the time and resources needed to get our results.

INTRODUCTION

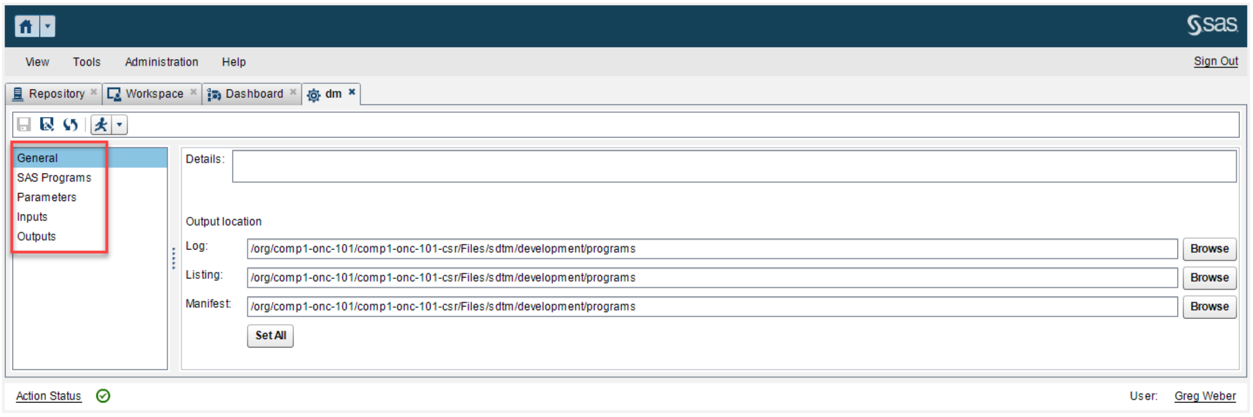
In LSAF (formerly known as SAS Drug Development), jobs are required in order to execute programs in the global Repository and permanently save the output and logs to the repository. The creation of jobs, while not overly complicated, is one of the more challenging and time-consuming activities for new users. It can be especially frustrating when your new job runs successfully in the Workspace, but then fails when run in the Repository. In this paper, we will review the components of a job, one of the common mistakes made by new users when creating jobs. We will review the execution of jobs and how this differs when running in the Workspace vs. the Repository. Once we have reviewed the basics around creating jobs, we will discuss how to use and take advantage of the Run and Populate feature to make the task of creating a job more foolproof and efficient.

JOBS IN LSAF - COMPONENTS OF A JOB

In LSAF, there are two working areas: the global Repository and the user's Workspace. You develop files in your workspace and then check the files in to the repository to share with other users. In order for a SAS program to be run in the repository and generate results accessible to all users, a job must be created.

To create a job, you define the following:

1. General Options: Description/Detail of the job and location of the log, listing and manifest.
2. SAS Programs: List of SAS programs to be run in the order to be executed.
3. Parameters: Parameters and values to pass to the job (converted to macro variables).
4. Inputs: Any input files and datasets.
5. Output: Folder(s) where the output data is saved.



Details:

Output location

Log: /org/comp1-onc-101/comp1-onc-101-csr/Files/sdtm/development/programs

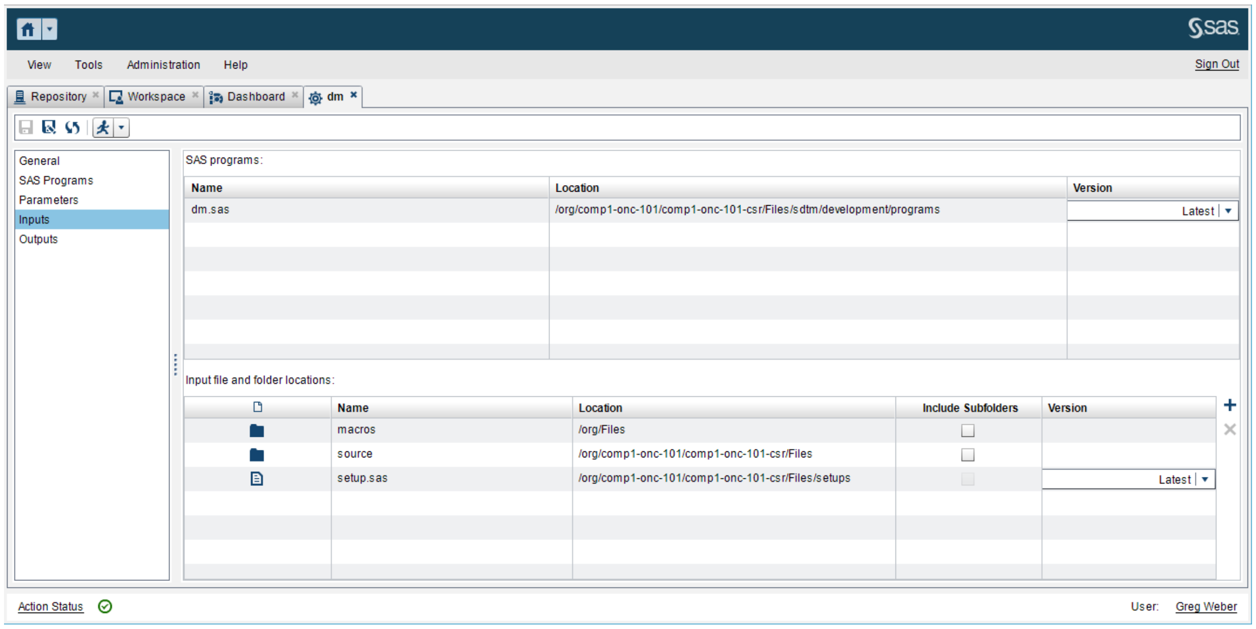
Listing: /org/comp1-onc-101/comp1-onc-101-csr/Files/sdtm/development/programs

Manifest: /org/comp1-onc-101/comp1-onc-101-csr/Files/sdtm/development/programs

User: Greg Weber

The screenshot above is the main job creation interface. In the red box are the items to define for each job. The focus of this paper is the definition of the job inputs.

When creating jobs in LSAF prior to version 4.7.1, all the inputs and outputs must be manually defined. The files and folders needed as inputs are browsed to and then added to the job. This is a tedious and error-prone process, which can cause frustration for users. Inputs are often missed when creating a job in the Workspace and then the job fails when run in the Repository. This leads to the user defining inputs at the folder level, which is easier and takes less time. The screenshot below shows, a job to create the DM dataset. Two of the inputs are located in the folders macros and source. These folders contain many files, but the user has chosen to define the input at the folder level instead of defining multiple inputs at the file level. The problem with this shortcut is that it results in defining many unnecessary inputs and creating inefficient jobs. We refer to jobs with unnecessary inputs as "bulky" jobs.



SAS programs:

Name	Location	Version
dm.sas	/org/comp1-onc-101/comp1-onc-101-csr/Files/sdtm/development/programs	Latest

Input file and folder locations:

	Name	Location	Include Subfolders	Version
📁	macros	/org/Files	☐	
📁	source	/org/comp1-onc-101/comp1-onc-101-csr/Files	☐	
🗎	setup.sas	/org/comp1-onc-101/comp1-onc-101-csr/Files/setups	☐	Latest

What is so bad about having a few extra inputs, you might ask?

Unlike when creating and running a job in the user Workspace, when a job runs in the Repository, the input and output folders and the input files are copied into the transient workspace (a temporary area created by LSAF when the job runs). If there are more inputs files than the job requires, then unnecessary data is copied to the transient workspace, which is a shared and limited storage resource. In addition, this increases job completion time. In the job displayed above, the user has defined the source folder as an input. When the job is run, it copies all the files in the source folder to the transient workspace. Though not obvious by looking at the inputs, the source folder for this project contains a large lab dataset that is not needed for this job.

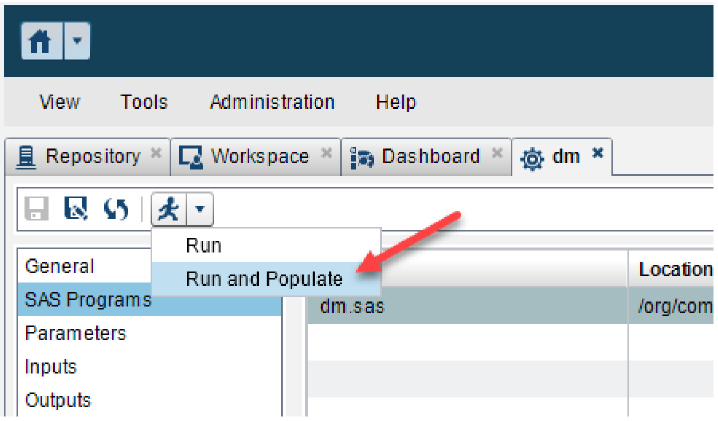
RUN AND POPULATE

Run and Populate is a new LSAF feature when creating and executing jobs in LSAF. Prior to this feature, all job inputs and outputs were defined manually as discussed above. Our contention is that job inputs should be defined at the file level (job outputs are always at the folder level). However, as discussed, defining to the file level is tedious and error-prone. This leads to the user defining inputs at the folder level, which is easier and takes less time. Run and Populate resolves this by automatically defining inputs only at the file level and only the files that are needed by the program, thus optimizing the job.

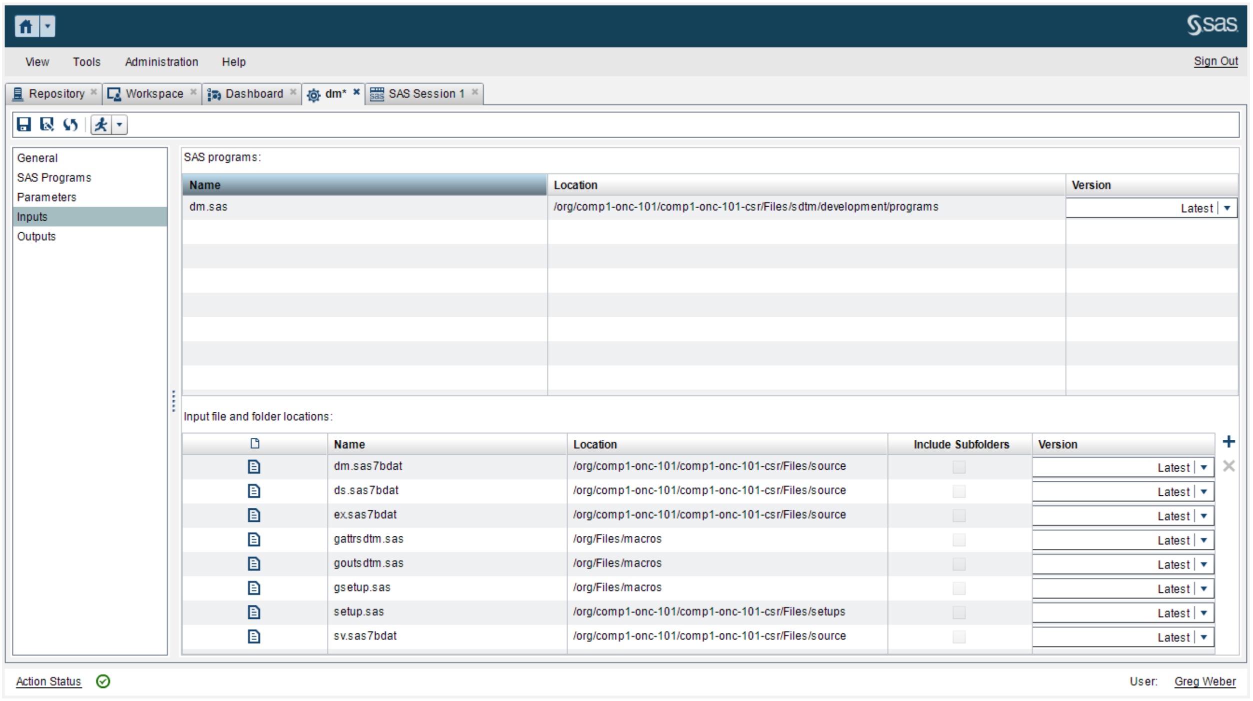
USING RUN AND POPULATE

1. As always, jobs must be created in the workspace.
2. User develops and successfully executes the SAS program(s) interactively in the workspace.
3. Programs(s) must run error-free to take advantage of Run and Populate.
4. Define the General Options, Parameters and SAS Program(s). These items are still defined manually.
5. Run the job selecting Run and Populate to define the inputs and outputs.
6. All inputs will be defined to the file level creating an efficient and optimally sized job.

The Run and Populate feature was added to the job creation interface.



The job above was manually created using Run and Populate with the following result. All inputs were automatically populated by LSAF and have been defined to the file level. Only required inputs were defined, resulting in a more efficient job.



SAS programs:

Name	Location	Version
dm.sas	/org/comp1-onc-101/comp1-onc-101-csr/Files/sdtm/development/programs	Latest

Input file and folder locations:

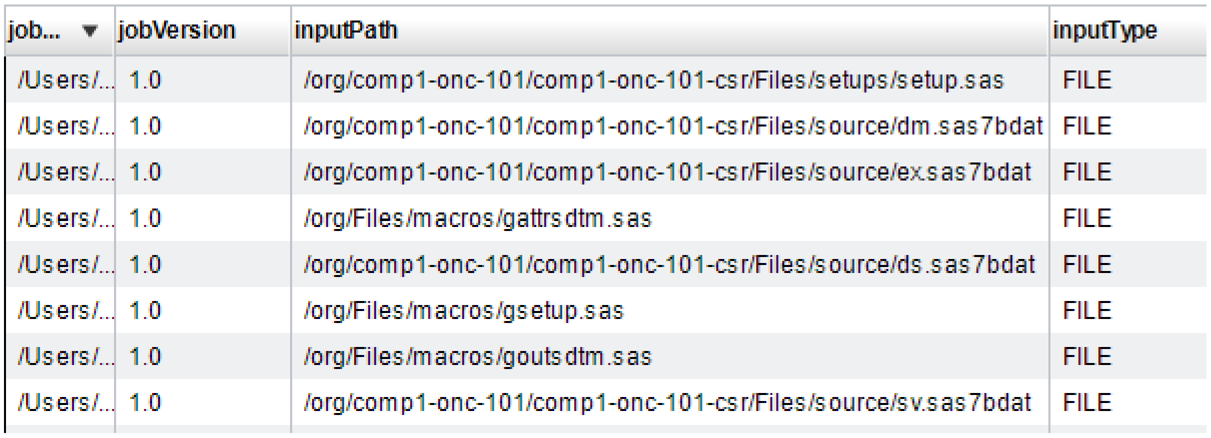
	Name	Location	Include Subfolders	Version
	dm.sas7bdat	/org/comp1-onc-101/comp1-onc-101-csr/Files/source		Latest
	ds.sas7bdat	/org/comp1-onc-101/comp1-onc-101-csr/Files/source		Latest
	ex.sas7bdat	/org/comp1-onc-101/comp1-onc-101-csr/Files/source		Latest
	gattrsdtm.sas	/org/Files/macros		Latest
	goutsdtm.sas	/org/Files/macros		Latest
	gsetup.sas	/org/Files/macros		Latest
	setup.sas	/org/comp1-onc-101/comp1-onc-101-csr/Files/setups		Latest
	sv.sas7bdat	/org/comp1-onc-101/comp1-onc-101-csr/Files/source		Latest

OTHER ADVANTAGES AND CONSIDERATIONS WHEN USING RUN AND POPULATE

When job inputs are defined to the file level (as they are when using Run and Populate), you can easily extract the job inputs from the job file. This is done by using the LSAF API macro %lsaf_getjobinputs:

```
%lsaf_getjobinputs(lsaf_path=./dm.job,sas_dsname=jobinputs);
```

Using this macro on the job shown above, produces the following dataset containing the input files with the full path.



job... ▼	jobVersion	inputPath	inputType
/Users/...	1.0	/org/comp1-onc-101/comp1-onc-101-csr/Files/setups/setup.sas	FILE
/Users/...	1.0	/org/comp1-onc-101/comp1-onc-101-csr/Files/source/dm.sas7bdat	FILE
/Users/...	1.0	/org/comp1-onc-101/comp1-onc-101-csr/Files/source/ex.sas7bdat	FILE
/Users/...	1.0	/org/Files/macros/gattrsdtm.sas	FILE
/Users/...	1.0	/org/comp1-onc-101/comp1-onc-101-csr/Files/source/ds.sas7bdat	FILE
/Users/...	1.0	/org/Files/macros/gsetup.sas	FILE
/Users/...	1.0	/org/Files/macros/goutsdtm.sas	FILE
/Users/...	1.0	/org/comp1-onc-101/comp1-onc-101-csr/Files/source/sv.sas7bdat	FILE

One need we had was the ability to copy a job from one LSAF analysis to another and then execute the job so we also needed to copy the input files to the new analysis. This is tedious to do manually so to automate this process, we created a macro to read a job, retrieve the inputs as shown above using the %lsaf_getjobinputs macro, iterate though the dataset and utilize the %lsaf_copy API macro to copy the required file to the new analysis. Other LSAF API macros used to make the process robust included %lsaf_objectexists and %lsaf_deleteobject to check if the file already exists and delete before copying.

This process works very well for us, but we did run into a couple of issues:

Since Run and Populate scans the SAS program to determine inputs and outputs for all programs run in the job, coding approach plays an important role. For example, we encountered a utility macro that was using proc contents on the whole library (_all_) to check the existence of one particular file. This resulted in run and populate identifying all files in the library as inputs. Clearly this led to an unnecessarily bulky job. The resolution was to recode the macro to not use _all_ and explicitly check for the existence of files. The lesson learned was that when designing our macros, we needed to take into account how this will affect the behavior of Run and Populate.

Another issue we encountered was with jobs that were designed to alter processing using job parameters. These jobs do not benefit from Run and Populate, since the inputs are dependent on the value of the chosen parameter.

Another consideration is for jobs that run multiple programs, for example, a job that runs all SDTM programs. In this case, it makes more sense to include inputs at the folder level instead of using Run and Populate, which will require the user to update the job each time a new input file is required.

CONCLUSION

The addition of the Run and Populate feature for creating LSAF jobs, has simplified the creation of efficient jobs in LSAF. Also, making jobs defined at the file level offers other advantages when automating processes that require the ability to extract the individual inputs from a job. While there are some considerations where Run and Populate is not always appropriate, the authors feel the functionality has great benefits.

REFERENCES

SAS® Life Science Analytics Framework 4.7: User's Guide

SAS® Life Science Analytics Framework - SAS Macro API, Version 1.5

ACKNOWLEDGMENTS

Steven Hege, Associate Director and Alexion SAS Administrator

Melissa Martinez, Principal Health and Life Sciences Industry Consultant ▪ Global Hosting and US Professional Services

RECOMMENDED READING

Hitchhiker's Guide to the Galaxy by Douglas Adams

CONTACT INFORMATION

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

Sonali Garg
Alexion Pharmaceuticals
475-230-3926
Sonali.Garg@alexion.com
http://www.alexion.com/

Greg Weber
DataCeutics
610-970-2333 (x226)
weberg@dataceutics.com
http://www.dataceutics.com/