



A Method for Automating Jobs in SAS® Life Science Analytics Framework (LSAF)

US PhUSE Connect 2019

Nicholas Masel
Janssen R&D

Drouville, Dragonfish
Drouville is a patient, graphic designer and
artist from Argentina who has survived
Multiple Myeloma and a relapse.

Agenda

- Introduction and scope
- Process overview
- Store the PROC SCAPROC output
- Parse the PROC SCAPROC output for inputs and outputs
- Plug the inputs and outputs into the job XML code and write out the job file
- Apply this process to one file our loop through multiple files

Introduction and Scope

Introduction and Scope

- Goal is to provide a method of job automation.
- Assumes knowledge of LSAF and jobs.
- Will not cover details of already well documented steps.
- Method outlined here is specific to one job per SAS program but can be modified for your needs.

Process Overview

Process Overview

1. Store the PROC SCAPROC output.
2. Parse the PROC SCAPROC output for inputs and outputs.
3. Plug the inputs and outputs into the job XML code and write out the job file.
4. Apply this process to one file or loop through multiple files.

Store the PROC SCAPROC output

Store the PROC SCAPROC output

```
%macro sca_outp (programPath=, LogFile=, scaProcOutput=);
```

```
  proc printto log="&LogFile";  
  run;
```

```
  filename scafile "&scaProcOutput" mod;
```

```
  proc scaproc;  
    write;  
  run;
```

```
  proc scaproc;  
    record scafile EXPANDMACROS;  
  run;
```

```
%include "&programPath";
```

```
proc scaproc;  
  write;  
run;
```

```
proc printto log=log;  
run;
```

```
%mend sca_outp;
```

PharmaSUG 2016 – Paper MS04

QA and Compliance Insights Using the SCAPROC Procedure

Ben Bocchicchio, SAS Institute, Cary NC
Sandeep Juneja, SAS Institute, Cary NC

ABSTRACT

The SCAPROC procedure is a new procedure that runs a SAS Code Analyzer. This SAS Code Analyzer captures metadata information about the contents of the SAS code that is run. It collects information on files that are used in the code's input, the code's output, and macro variables used **while the code is running**.

How does this procedure relate to quality assurance (QA) and compliance? Just imagine if you could collect all the metadata about all the SAS code that is run to generate output for a FDA submission. You could programmatically prove that all the data referenced for input is consistently used, all macros called in the programs are consistent (without worrying about calling a generic macro when a project-specific macro is required), and determine that all designated output is saved to the correct location. Would this make you feel more confident about your submission? This presentation reveals these uses of SCAPROC. In addition, this presentation discusses other potential uses of this information.

INTRODUCTION

The SCAPROC procedure implements the SAS Code Analyzer, which captures information about input, output, and the use of macro symbols from a SAS program while it is running. The SAS Code Analyzer can write this information and the information that is in the original SAS file to a file that you specify. When developing SAS code for clinical trial analysis, ensuring the correct data sets are used as input is paramount to creating consistent output. The output can be either the submission data sets or the supporting Figures, Tables, and Listings. SAS programmers have developed various process to ensure that the correct data is chosen; however, issues can still arise. The PROC SCAPROC procedure can be another tool to help verify that SAS programs are producing consistent results. The output can be used to perform impact analyses. The PROC SCAPROC procedure first appeared in Release 9.2 of Base SAS® Software.

PROC SCAPROC SETUP

To produce the PROC SCAPROC output, a simple macro can be developed that only needs three inputs, the SAS program to analyze, the output Log location, and the location to store the SCAPROC results as shown in Figure 1.

Figure 1: Simple Macro for PROC SCAPROC Output

```
%macro sca_outp (programPath=, LogFile=, scaProcOutput=);  
  
  proc printto log="&LogFile"; run;  
  
  filename scafile "&scaProcOutput" mod;  
  
  proc scaproc; write; run;  
  
  proc scaproc; record scafile EXPANDMACROS; run;  
  
  %include "&programPath";  
  
  proc scaproc; write; run;  
  
  proc printto log=log; run;  
  
%mend sca_outp
```

This macro can be called from list of all SAS programs for a project or protocol.

**Parse the PROC SCAPROC output
for inputs and outputs**

Parse the PROC SCAPROC output for inputs and outputs

- Full code found using the link to the right
- PRX parse using keywords and patterns

The screenshot shows the SAS Support website. At the top, there's a navigation bar with the SAS logo, the tagline "THE POWER TO KNOW", a "Sign In" link, and a search bar. Below this is a secondary navigation bar with links for "Home", "Support", "Learn", and "Connect". The main content area is titled "SUPPORT / SAMPLES & SAS NOTES". On the left, there's a "SUPPORT" sidebar with various links like "Submit a Problem", "Check Problem Status", "SAS Administrators", "Security Bulletins", "License Assistance", "Manage My Software Account", "Downloads & Hot Fixes", and "Samples & SAS Notes". The "Samples & SAS Notes" section is expanded, showing sub-links like "Browse by Topic", "Search Samples", "Search Usage Notes", "Search Installation Notes", and "Search Problem Notes". The main content area displays "Sample 58047: SAS® Life Science Analytics Framework - Parse output from PROC SCAPROC to create a data set with inputs and outputs". It has tabs for "Details", "Full Code", "Downloads", "About", and "Rate It". The "Overview" section states: "This sample code parses the output from PROC SCAPROC and creates a data set with information about the program that was run. Information included in the data set:" followed by a bulleted list: "Program name", "Inputs", "Outputs", and "Macros used". The "Additional Documentation" section has a link for "Overview: SCAPROC procedure". At the bottom, there's a disclaimer: "These sample files and code examples are provided by SAS Institute Inc. 'as is' without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Recipients acknowledge and agree that SAS Institute shall not be liable for any damages whatsoever arising out of their use of this material. In addition, SAS Institute will provide no support for the materials contained herein." The footer of the page features a dark blue background with four columns of links: "Connect" (Blogs, Contact, Events, News, RSS), "Customer Support" (Certification, Knowledge Base, Documentation, SAS Books, Training), "Insights & Trends" (Analytics, Big Data, Business Intelligence, Data Management, Fraud & Security), and "Quick Links" (Careers, How to Buy, White Papers, Webinars, Resource Center).

Plug the inputs and outputs into the job XML code and write out the job file

Plug the inputs and outputs into the job XML code and write out the job file

- Create job template
 - Setup a default job that addresses repeated job assignments
 - For example, environment setup program, where to write log, lst, and manifest, required parameters for every job
 - Add placeholder to plug in inputs, outputs, program name, parameter values
- Replace some template placeholders with macro variables
- Build XML syntax for input and outputs
 - Use results from PROC SCAPROC parse
 - Merge onto job template

Job Template

```
<?xml version="1.0" encoding="UTF-8"?>
<job releaseVersion="1.0" description="" executionMode="SEQUENTIAL" LSAFVersion="4.0" log="./log" lst="./lst"
mnf="./mnf">
  <tasks>
    <task path="./programs/envsetup.sas" />
    <task path="./programs/PROGRAMPLACEHOLDER.sas" />
  </tasks>
  <taskSpecs>
    <inputSpec path="./programs/envsetup.sas" type="FILE" version="*" includeSubFolders="false" />
    <inputSpec path="./programs/PROGRAMPLACEHOLDER.sas" type="FILE" version="*" includeSubFolders
  </taskSpecs>
  <otherSpecs>
    <XMLINPUTPLACEHOLDER>
  </otherSpecs>
  <outputPaths>
    <XMLOUTPUTPLACEHOLDER>
  </outputPaths>
  <outputSpec enableVersioningForNewFiles="true">
  <versionType type="MAJOR" />
  </outputSpec>
  <parameters>
    <parameter name="source_ds" type="CHARACTER" defaultValue="SOURCEDSPPLACEHOLDER" label="Source data" />
    <parameter name="source_meta" type="CHARACTER" defaultValue="SOURCEMETAPLACEHOLDER" label="Source
metadata" />
  </parameters>
</job>
```

Replace some template placeholders with macro variables

```
data tempjob;  
  infile "&_SASWS_/testFolder/jobs/templateJob.sas" length=len;  
  input string $varying32767. len;  
  string = tranwrd(string, "PROGRAMPLACEHOLDER", "&&file&jobnumber");  
  string = tranwrd(string, "SOURCEDSPPLACEHOLDER", "&source_ds");  
  string = tranwrd(string, "SOURCEMETAPPLACEHOLDER", "&source_meta");  
  ord=_n_;  
  string=strip(string);  
  put string;  
run;
```

Build XML syntax for input and outputs

```
data feedToJob;
set scaResults;

length string $32767;

if lowercase(type) = "input" then do;
  string = "<XMLINPUTPLACEHOLDER>";
  xml = '<inputSpec path=""|
        strip(tranwrd(file_path,"&_SASWS_","")) ||
        "' type="FILE" version="*" includeSubFolders="false" />';
end;

if lowercase(type) = "output" then do;
  string = "<XMLOUTPUTPLACEHOLDER>";
  xml = '<outputPath path="" ||
        tranwrd(substr(file_path , 1, find(file_path, '/', -length(file_path))-1) ,"&_SASWS_" , "")||
        "' includeSubFolders="false" />';
end;

run;
```

Merge together and write job file

```
data finaljob;  
merge tempjob  
    feedToJob(in=b);  
  
    by string;  
  
    if b then string = strip(xml);  
  
run;  
  
filename final "&_SASWS_/testFolder/jobs/&&file&jobnumber...job";  
  
data _null_;  
    file final;  
    set finaljob(keep=string);  
    put string;  
run;
```

**Apply this process to one file or
loop through multiple files**

Apply this process to one file or loop through multiple files

- Wrap process in a loop to create multiple jobs
- Add macro to return contents of a directory
- Piece together tool to create one job per SAS program found in a directory
- Code can be tweaked to also allow multiple programs per job if needed



Thank you

Nicholas Masel
nmasel@its.jnj.com

Drouville, Dragonfish
Drouville is a patient, graphic designer and artist from Argentina who has survived Multiple Myeloma and a relapse.

Q&A

janssen

PHARMACEUTICAL COMPANIES OF

Johnson & Johnson

