PharmaSUG 2023 - Paper RW-154

Automating Non-Standard New Study Set Up with a SAS® Based Work Process

Bo Zheng, Li Ma, and Xingshu Zhu, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

It's essential and effective to start a new study utilizing broad institutional standard macros, especially for data quality review (DQR) and analysis and reporting (A&R) in real-world-evidence (RWE) studies. However, the rapid developmental pace of studies introduces the challenge of effectively managing standard macros and often requires the identification of what standard macros are needed and where they are located. Having a clearly defined work process and tool for automatically copying specific macros and folders to the new study directory significantly reduces the time spent on manual work and improves reliability and efficiency across multiple studies. This paper highlights the functional portions of a real-life work process and a SAS macro to easily automate this task while preserving the original file timestamps for traceability.

INTRODUCTION

The increasing popularity of RWE studies and the rapid development of customized macros and code libraries have created new challenges for organizations. Macros and code snippets may reside in multiple separate folders, leading to difficulties in copying these helpful programs over to a new study. This can be a time-consuming and complex process, with end-users often having to manually copy files from multiple subfolders into the new or desired study folder. In this paper, we defined a clear work process and developed a SAS macro to automatically aggregate and copy user-specified files into a new or existing study directory. We've found that this approach made our project work more consistent, improved reliability and efficiency across multiple new studies, and helped to improve the onboarding experience for new staff members.

DEFINE THE GENERAL WORK PROCESS

Defining a work process involves several important considerations such as the type of analysis platform, its operating system, and the software and hardware involved. Consistency is key when organizing folders, and it's recommended to use a uniform folder structure across studies that also include clearly defined subfolders for documents, programs, and macros. This example approach can help to streamline existing workflows and to improve collaboration within and across teams.

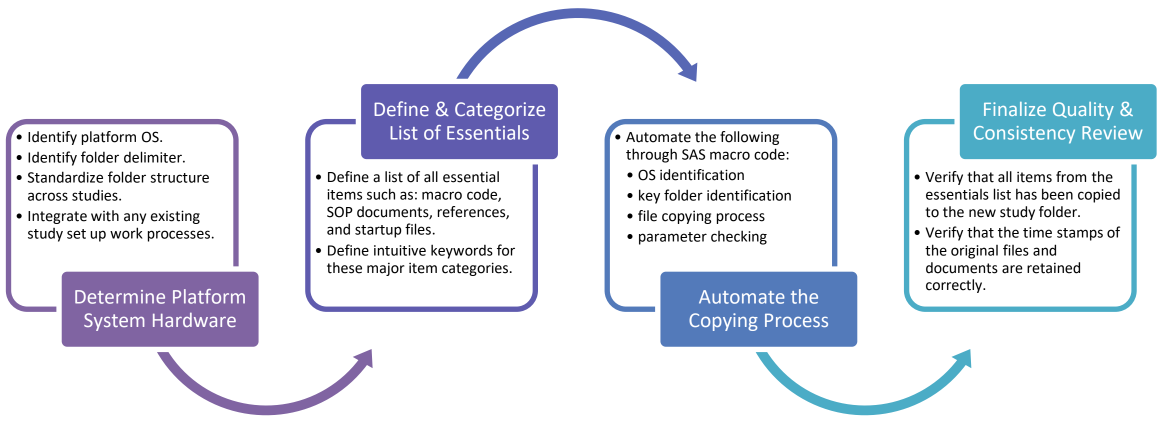

Determine Platform System Hardware
• Identify platform OS.
• Identify folder delimiter.
• Standardize folder structure across studies.
• Integrate with any existing study set up work processes.
Define & Categorize List of Essentials
• Define a list of all essential items such as: macro code, SOP documents, references, and startup files.
• Define intuitive keywords for these major item categories.
Automate the Copying Process
• Automate the following through SAS macro code:
• OS identification
• key folder identification
• file copying process
• parameter checking
Finalize Quality & Consistency Review
• Verify that all items from the essentials list has been copied to the new study folder.
• Verify that the time stamps of the original files and documents are retained correctly.

Figure 1. Suggestions for Defining the General Work Process

To ensure a smooth transition, start by mapping out a list of essential program codes, documents, and folder locations needed for the new study, including any traditionally manually copied files. File extensions and subfolder names are generally useful to identify for filtering out nonrelevant files from the copying process. Categorizing groups of similar files to be passed as SAS macro parameter inputs can also improve the automation planning process, especially for different types of files located in different folders, for example: “MACROS”, “STARTUP”, or “SOPDOCS”.

Once the list of critical files and their folder locations have been finalized, it’s time to plan an automated process for copying the list of files from source to destination with a simple SAS macro. This paper covers SAS automation examples for Windows and Unix/Linux-based systems but most of the general code and logic can be applied to other operating systems. It’s recommended to also define a set of post-run quality and consistency review requirements and checks at the group or organization level. Additional functionality, such as automatically generating a checklist of files copied for manual review, can also be added to the final process.

The general automation needs for our work group are defined below:

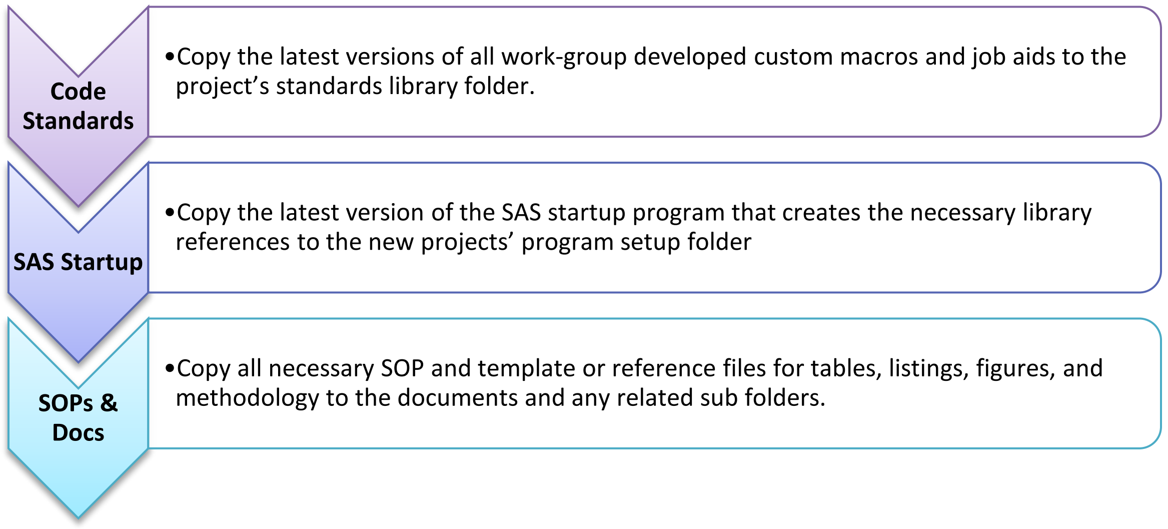

Code Standards
•Copy the latest versions of all work-group developed custom macros and job aids to the project’s standards library folder.
SAS Startup
•Copy the latest version of the SAS startup program that creates the necessary library references to the new projects’ program setup folder
SOPs & Docs
•Copy all necessary SOP and template or reference files for tables, listings, figures, and methodology to the documents and any related sub folders.

Figure 2. Priority automation components of example work process

AUTOMATE THE WORK PROCESS IN SAS

After defining the list of essential files to copy, their locations, and any operating system requirements, the next step is automating the copying process in SAS. For easier debugging, the provided example macro covers one folder/directory and its file contents per call, though it can be modified with additional code to handle more than one at a time if the file type, category, and input and output locations are consistent.

Macro parameters that define the source, destination, file extension, and additional filtering and debugging options can be used to enhance overall functionality:

SAS Macro Parameter	**Description**
COPY_FILETYPE	Required; a predefined keyword that can be used to designate the file category to copy. Examples: “MACROS”, “STARTUP”, or “SOPDOCS”.
SOURCEPATH	Required; source file or folder location path to copy from.
DESTPATH	Required; destination folder location path to copy to.
FILTER	Optional; this parameter can be used to filter out subfolders that do not need to be copied, such as backups.

SAS Macro Parameter	Description
LOGPATH	Optional; this parameter can be used to specify where the external log is saved, otherwise it will be saved in the same folder defined by DESTPATH in the example code.
DEBUG	Optional; this parameter can be used to delete or retain intermediate datasets for debugging in the example code.

Table 1. Suggested parameters for example SAS automation macro

EXAMPLE SAS MACRO & USAGE

Creating the example automation macro with the recommended parameters from table 1:

```
%macro example0macro(copy_filetype= ,sourcepath= ,destpath=
                    ,filter= ,logpath=, debug=N);
```

Using the correct delimiter when separating different elements in a command line or file path argument is important, as delimiters can be operating system specific. In Windows, the delimiter is “\”, while in Linux, it's “/”. Using the wrong delimiter can cause errors in the SAS macro and prevent the process from returning the correct results.

It's recommended to create a macro variable (&DELIM) to store the delimiter. The example code below also outlines optional enhancements to perform additional automated preparation tasks for each of the three path variables "sourcepath", "destpath", and "logpath":

- Replace any forward slashes with backward slashes if the operating system is "WIN".
- Replace any backward slashes with forward slashes if the operating system is "LIN X64".
- Remove the trailing slash from each path variable if one exists.

An abbreviated example of adding automatic preparation functionality to “sourcepath”:

```
%global delim;
%if %index('WIN',&sysscp)>0 %then %do;
%let delim=\;
  %let
sourcepath_temp=%sysfunc(tranwrd(%bquote(&sourcepath),%str(/),%str(\)));
    /* Check if directory ends with a slash, get rid of it if exists*/
    %if %substr(%sysfunc(strip(%sysfunc(reverse(&sourcepath_temp.)))),1,1)=\
    %then %do;
      %let sourcepath=%substr(&sourcepath_temp., 1,
      %eval(%length(&sourcepath_temp.)-1));
    %end;
    %else %do;
      %let sourcepath=&sourcepath_temp.;
    %end;

    ... /* use a similar technique to define destpath and logpath */

    %end;
%else %if %index('LIN X64',&sysscp)>0 %then %do;
  %let delim=/;

    ... /* use a similar technique for function on Linux on 64 platforms */

  %end;
```

This recommended section of SAS macro code retrieves a list of files in a specific directory (&sourcepath) and uses various SAS functions that interact with the operating system (filename, dopen, dread, and dclose) to create a new dataset called 'macpath':

```
/* Create dataset to store listing of files and attributes */
data macpath(keep=num file_name file );
  length file_name $100. file $80.;
  rc=filename('dir',"&sourcepath.");
  dirid=dopen('dir');
  do i=1 to dnum(dirid);
    file_name=dread(dirid,i);
    file=upcase(scan(file_name, 1, '.'));
    file_extension=substr(file_name, index(file_name, '.') + 1);
    num=i;
    output;
  end;
  rc=dclose(dirid);
run;
```

The code also includes an optional filter that can exclude certain files as defined by the &filter macro variable:

```
/* Extra filtering of files if needed */
%if %length(&filter.) > 0 %then %do;
  data macpath;
    set macpath;
    where file not in (%upcase(&filter.)) and file_extension not in
(&filter.);
  run;
%end;
```

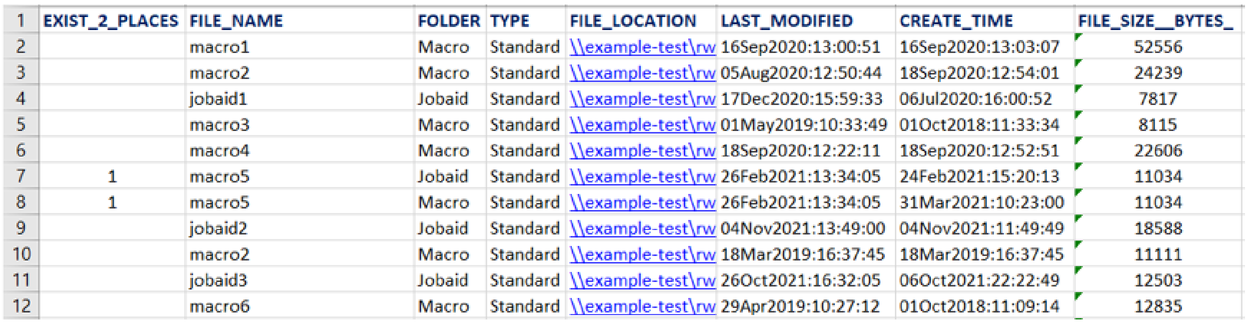
The total count of files that are copied is stored in a macro variable:

```
/* Total count of files */
proc sql;
  select count(distinct file_name) into: totcnt separated by ''
  from macpath
quit;
%put fff=&totcnt.;
```

The ‘macpath’ dataset is further refined using additional functions (filename, fopen, foptname, and finfo) to aggregate the results and retrieve additional helpful details like file creation date, modified date, and file size:

```
/* Create dataset for listing report of files only */
%let i=0;
data info_;
  length infoname infoval $600;
  do i = 1 to &totcnt.;
  set macpath;
    /* file_name_temp definition depends on platform folder structure */
    file_name_temp="&sourcepath."||"&delim."||strip(file_name);
    rc=filename("temp",file_name_temp);
    fid=fopen("temp");
    if (fid > 0) then do;
      infonum=foptnum(fid);
        do k=1 to infonum;
        infoname=foptname(fid,k);
        infoval =finfo(fid,infoname);
        output;
      end;
      close=fclose(fid);
    end;
    rc=filename("temp",'');
  end;
  drop rc fid;
run;
```

Finally, the intermediate dataset “info_” is sorted and transposed by “file_name” to produce the output “info_ready” summary dataset. The “macpath” and “info_ready” datasets are useful for quickly creating a list of files in a directory for quality and consistency review.



1	EXIST_2_PLACES	FILE_NAME	FOLDER	TYPE	FILE_LOCATION	LAST_MODIFIED	CREATE_TIME	FILE_SIZE__BYTES_
2		macro1	Macro	Standard	\\example-test\rw	16Sep2020:13:00:51	16Sep2020:13:03:07	52556
3		macro2	Macro	Standard	\\example-test\rw	05Aug2020:12:50:44	18Sep2020:12:54:01	24239
4		jobaid1	Jobaid	Standard	\\example-test\rw	17Dec2020:15:59:33	06Jul2020:16:00:52	7817
5		macro3	Macro	Standard	\\example-test\rw	01May2019:10:33:49	01Oct2018:11:33:34	8115
6		macro4	Macro	Standard	\\example-test\rw	18Sep2020:12:22:11	18Sep2020:12:52:51	22606
7	1	macro5	Jobaid	Standard	\\example-test\rw	26Feb2021:13:34:05	24Feb2021:15:20:13	11034
8	1	macro5	Macro	Standard	\\example-test\rw	26Feb2021:13:34:05	31Mar2021:10:23:00	11034
9		jobaid2	Jobaid	Standard	\\example-test\rw	04Nov2021:13:49:00	04Nov2021:11:49:49	18588
10		macro2	Macro	Standard	\\example-test\rw	18Mar2019:16:37:45	18Mar2019:16:37:45	11111
11		jobaid3	Jobaid	Standard	\\example-test\rw	26Oct2021:16:32:05	06Oct2021:22:22:49	12503
12		macro6	Macro	Standard	\\example-test\rw	29Apr2019:10:27:12	01Oct2018:11:09:14	12835

Display 1. Example of customized review listing created from “info_ready” summary dataset

The SAS code below first checks if the macro parameter “copy_filetype” is defined as "ANY" (any file type) using the %if statement and what system type is being used with the %index function:

```
/* Example to copy contents of any single file or folder */
%if %upcase(&copy_filetype.)=ANY %then %do;
```

The essential commands “XCOPY” and “CP -P” can also be used in conjunction with SAS macro variables that are user-created or automatically generated by the SAS system to enhance their versatility, for example:

- &SYSSCP is an automatic variable that can identify the operating system.
- &SYSSCPL can be used for further refinement for specific operating system versions or bitness.

The value of &SYSSCP will depend on the computing platform being used:

- Windows-based platforms will have a value of 'WIN'.
- Linux on 64 will have a value of 'LIN X64'.
- z/OS will have a value of 'OS'.

If the system is Windows, the “&filter” parameter won’t work with this simple example without Windows batch to only copy specific files defined in the “macpath” dataset. Alternatively, replace the “&sourcepath” wild cards (*.*) to control the file name or extension to be copied.

For general applications, use the %sysexec statement to execute the XCOPY command to copy the files from the user-defined source path to the destination path:

```
  %if %index('WIN', &sysscp)>0 %then %do;
    %sysexec xcopy "&sourcepath.\*.*"  "&destpath.\*.*" ;
  %end;
```

A more complex example could involve copying over standard code from multiple source folders, then this general example using Windows batch processing can be useful:

```
options noxwait mprint;
data _null_;
  file "C:\Users\&SYSUSERID\macrocopy_example.bat" LRECL=2000;
  /* Exact folder and delimiter structure will vary based on application */
  %do i = 1 %to &totcnt.;
    put 'xcopy /k /r /y' ' "' "&&srcfile&i." '"' ' "' "&destpath." '"' `
        >> ' '"' "&logpath.\macrocopy_logfile.txt" '"';
  %end;
run;

/* Executing the .bat file */
%sysexec C:\Users\&SYSUSERID\macrocopy_example.bat;
```

If the system is a Linux system, the code first filters out any directory names in the source path and then recreates the dataset "macpath" containing the remaining file names and extensions in the source path:

```
%else %if %index('LIN X64', &sysscp)>0 %then %do;
  data macpath;
    set macpath;
    where index(file_name, '.') > 0 ;  /* exclude folders */
  run;
```

The code then creates a series of macro variables to hold the total number of files and the source and destination paths for each file to be copied:

```
  data _null_;
    set macpath end=eof;
    call symput('srcfile'||compress(put(_n_,3.))
                ,cats("&sourcepath./", strip(file_name)));
    call symput('destfile'||compress(put(_n_,3.))
                ,cats("&destpath./", strip(file_name)));
    if eof then call symput('totcnt', compress(put(_n_,3.)));
  run;
  %put srcfile1=&srcfile1 srcfile2=&srcfile2 totcnt=&totcnt;
  %put destfile1=&destfile1 destfile2=&destfile2;
```

Finally, the %do loop uses the %sysexec statement to execute a Unix/Linux shell command “CP -P”, which is used to copy files within a specified directory with the option to preserve their permissions and other attributes such as timestamps and ownership:

```
  %do j = 1 %to &totcnt.;
    %sysexec cp -p "&&srcfile&j."  "&&destfile&j.";
  %end;
%end;
```

This simplified SAS example can be enhanced for complex applications such as pulling from multiple code repositories using Windows batch files or Linux shell scripts if the platform-specific security and user permissions are enabled.

DISCUSSION

The example macro codes provide a comprehensive overview for defining and automating a work process for copying files to a new or existing study folder. We found that utilizing an existing standardized folder structure, program naming convention, and startup program, alongside the organization or group's standard operating protocol (SOP) made the entire process much more intuitive.

It also eliminated the risk of human error during the copying process and ensured that all important files were included. We did test multiple other copy methods but found that on Windows “XCOPY” and on Unix/Linux “CP -P” were the most consistent methods to copy files while also retaining their permissions, original creation timestamps, and other important system attributes.

CONCLUSION

Implementing the proposed work process and the automated SAS copy macro:

- Significantly improved reliability and efficiency across multiple new studies.
- Enhanced the onboarding experience for new staff members and contractors by reducing the time they spent searching for templates and developing redundant analysis code.
- Eliminated the risk of manual copy/paste errors and poor version control issues with a seamless and reliable method to move files from multiple repositories to their correct destinations.

Overall, we found that automating this work process in SAS significantly streamlined our existing workflows and noticeably improved collaboration within and across our various teams, and improved the consistency, quality, and compliance of RWE and other non-standard studies with our organization's broader standards and SOPs.

ACKNOWLEDGMENTS

The authors would like to thank their real-world evidence and epidemiology colleagues at Merck & Co., Inc., Upper Gwynedd, PA, USA, for their partnership in support of this work.

CONTACT INFORMATION

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

Bo Zheng
Merck & Co. Inc.
351 N Sumneytown Pike
North Wales, PA 19454, USA
bo.zheng1@merck.com

Li Ma
Merck & Co. Inc.
351 N Sumneytown Pike
North Wales, PA 19454, USA
li.ma2@merck.com

Xingshu Zhu
Merck & Co. Inc.
351 N Sumneytown Pike
North Wales, PA 19454, USA
xingshu_zhu@merck.com

TRADEMARK