Confidential

PharmaSUG 2023 - Paper QT-152

A utility tool to assist with the setup of the startup environment for remote access

William Wei, Shunbing Zhao, Merck & Co., Rahway, NJ, USA

ABSTRACT

Along the course of the clinical trial process, there are multiple deliverables for different purposes such as interim analyses, DMC, safety evaluations, etc. These will require programming updates for each deliverable, which will reside in multiple subfolders and may involve multiple programmers. Therefore, the study level information also needs to be updated in the programming environment settings, for example the data cut date, study level macro variable values for titles/footnotes, folder structures, etc. Usually, programmers keep many copies of startup/setup files for developer/validation purposes and for different SAS® platforms. With this approach, it takes considerable effort to keep track of updates that are done in the main setup file. It is tedious and error prone. If we can maintain one central copy of the setup file, and the other setup files can call the central setup file, it will reduce the effort to track the many setup file updates. In the remote submit mode, some techniques which SAS users do not use often need to be applied to pass SAS macro variables from local machine to server. In this paper, we are sharing the techniques used to pass the local macro variables to the server in order to setup the startup file. Also, an example macro and walk through of implementation and use are provided in this paper.

INTRODUCTION

When we are working on a deliverable, there are many programmers working on different ADaM datasets and TFLs, performing development and validation. There is common information that is shared with all the programmers – study environment/deliverable level, for example, the data cut-off date, common footnotes, etc. In the current settings, there are many copies of deliverable level settings saved in the system. Multiple programmers are maintaining their own copies (Figure 1). In the case of remote submit, users use PC SAS and remotely submit SAS code to the server using the command “rsubmit”. In this case, developers shared one setup file, because the outputs are designated to one final folder. The validation programmers will maintain their own copies because they are working in different folders. The same parallel setup occurs in server submit situations. For example, the users can run their SAS code using SAS EG which is directly connected to SAS server.

POTENTIAL ISSUE RAISED

Maintaining multiple copies of setup files in each deliverable can become tedious, especially when updates such as data cutoffs, common titles/footnotes, and other deliverable-level information need to be applied across all copies (Figure 1). If one programmer forgets to update the setup, it may cause output to be wrong, or result in validation failure. It is easy to fix, but many times, the issue could not be identified, especially when we have time constraints.

SOLUTION

Instead of keeping many copies of setup files for a deliverable, the protocol lead only maintains one “core setup” file. To ensure that all programmers working on the same deliverables have the most current environment setup, they will call the 'core setup' file. This eliminates the need for individual programmers to update their own setup files and allows them to focus on programming development (Figure 2).

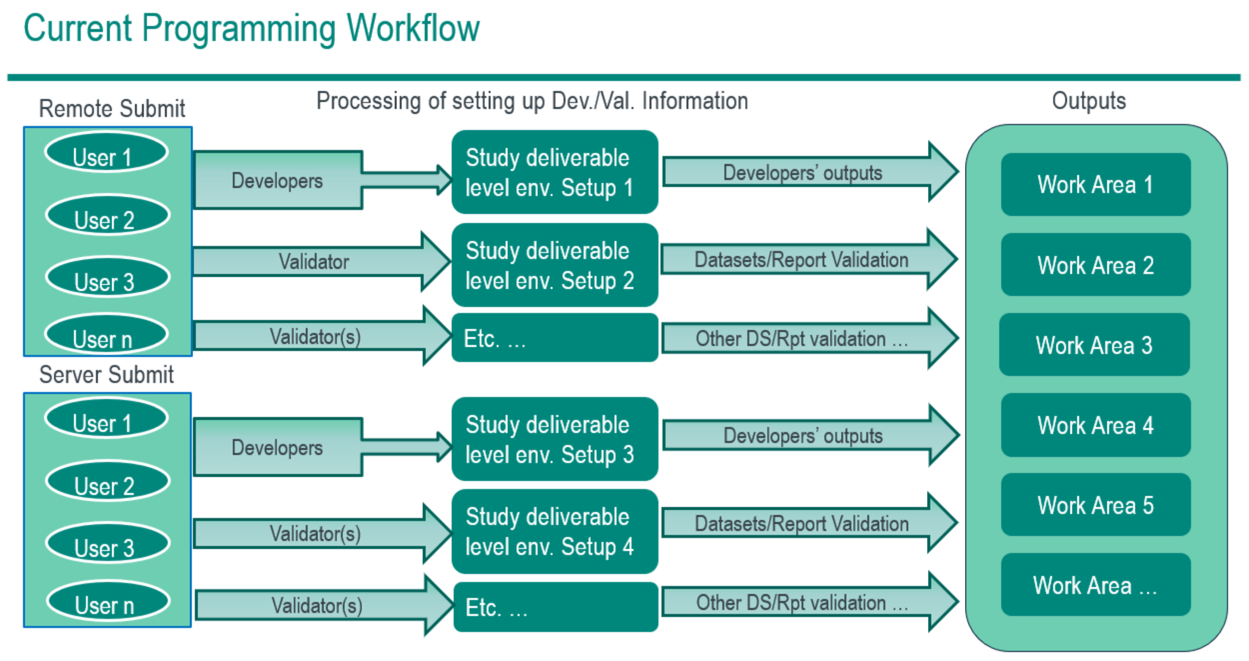

Current Programming Workflow
Remote Submit
User 1
User 2
User 3
User n
Server Submit
User 1
User 2
User 3
User n
Processing of setting up Dev./Val. Information
Developers
Validator
Validator(s)
Developers
Validator(s)
Validator(s)
Study deliverable level env. Setup 1
Study deliverable level env. Setup 2
Etc. ...
Study deliverable level env. Setup 3
Study deliverable level env. Setup 4
Etc. ...
Developers' outputs
Datasets/Report Validation
Other DS/Rpt validation ...
Developers' outputs
Datasets/Report Validation
Other DS/Rpt validation ...
Outputs
Work Area 1
Work Area 2
Work Area 3
Work Area 4
Work Area 5
Work Area ...

Figure 1. Current settings for Analysis & Reporting deliverables.

Remote submit: User submit SAS program using PC SAS and remotely submit code to SAS server. The developers share a common setup file, and validation programmers have their own setup files which are maintained by individual programmer.

Server submit: User submit SAS program directly to server, such as using SAS EG. Same as the remote submit, the developers share a common setup file, and validation programmers have their own setup files which are maintained by individual programmer.

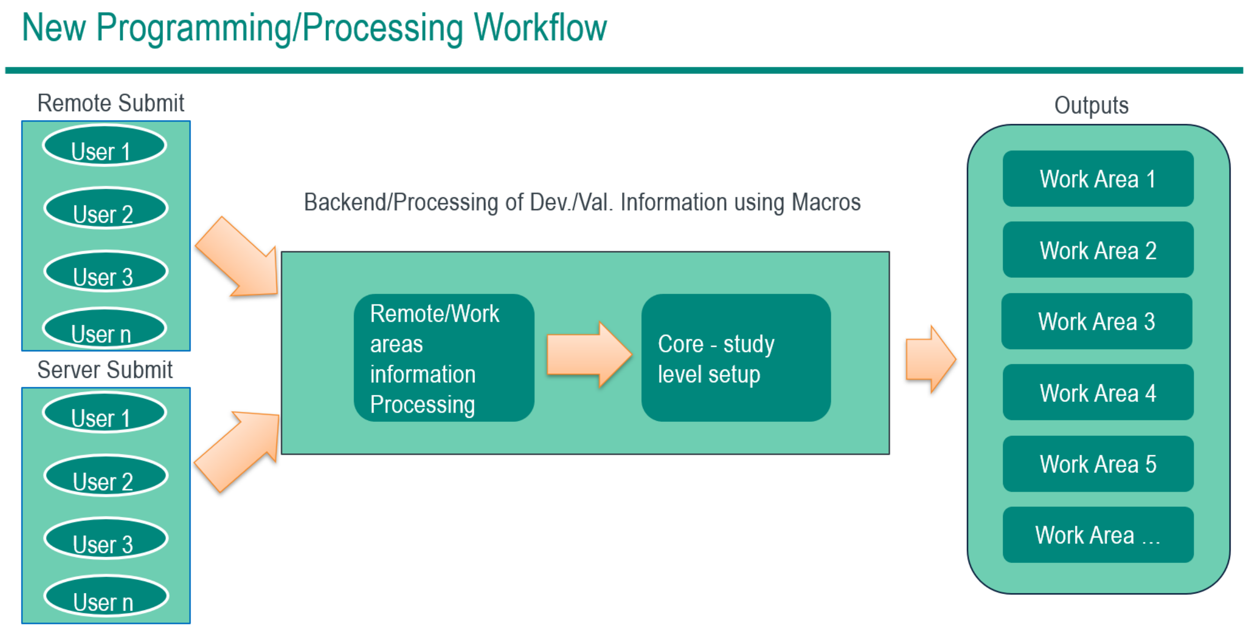


Figure 2. New settings for Analysis & Reporting deliverables.

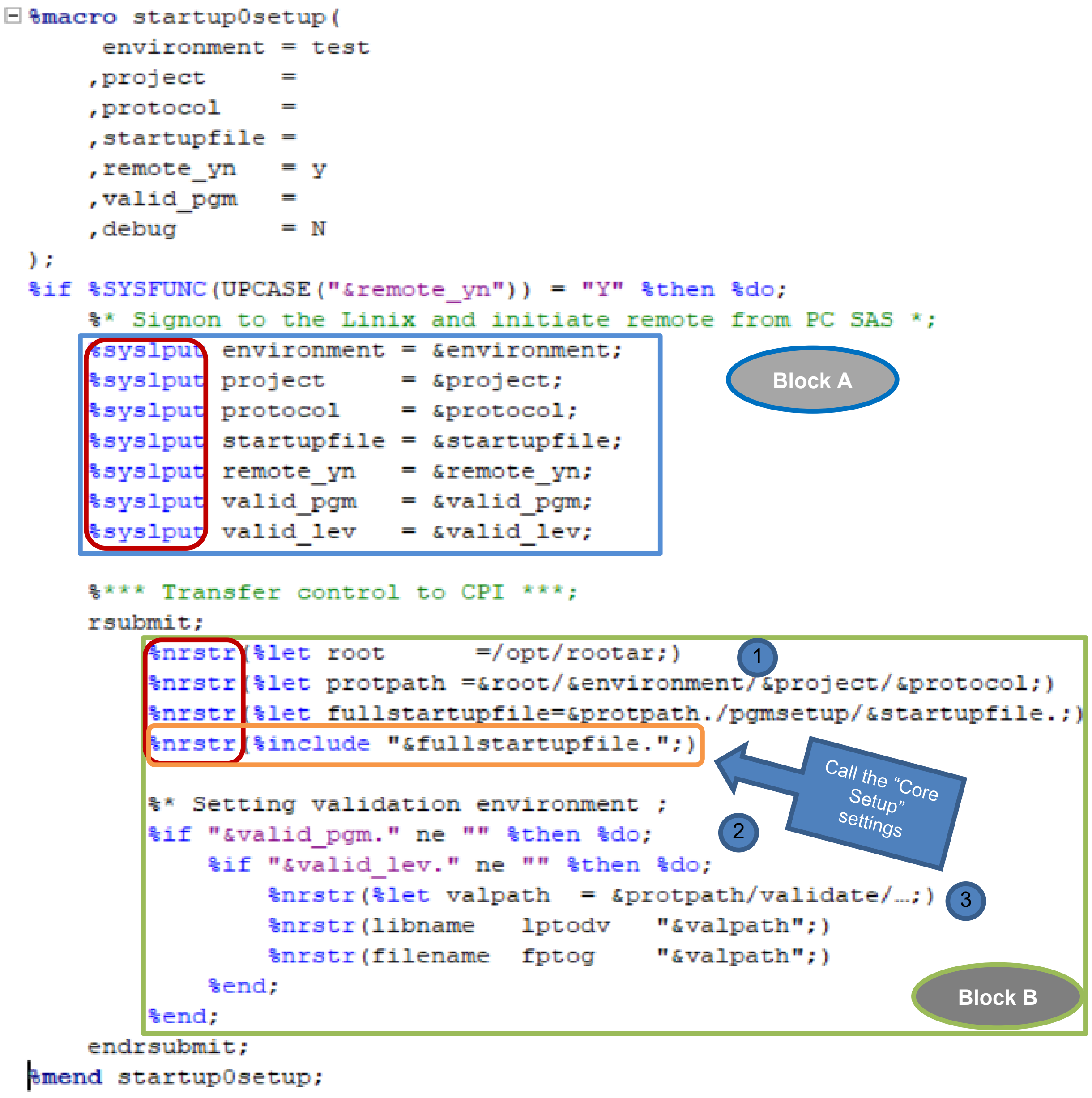
A macro was developed to call the “core setup file” and process the information of “Remote” or “Server” submit, develop or validation environment, and assign working area accordingly. This will save programmers’ precious time and will be less error prone.

KEY TECHNIQUES OF DEVELOPING THE MACRO

1. Assigning proper working environment according to user's inputs
2. Techniques for passing local macro variables to remote hosts

The first technique - assigning proper working environment is relying on our standard folder structures. Based on the provided details such as the purpose of development/validation, the name of the dataset or report, we can setup macro variable or library name in accordance with our standard folder structure design.

In this paper, we focus on the 2nd SAS technique – "Using the Macro Facility with SAS/CONNECT"1, which will transfer information from local machine to SAS server for "remote submit" users. Please refer to the following macro call (Figure 3), and we will use this simple macro call to illustrate the transfer of macro variables from local to server.



```
%macro startup0setup(
      environment = test
     ,project     =
     ,protocol    =
     ,startupfile =
     ,remote_yn   = y
     ,valid_pgm   =
     ,debug       = N
);
%if %SYSFUNC(UPCASE("&remote_yn")) = "Y" %then %do;
    %* Signon to the Linix and initiate remote from PC SAS *;
    %syslput environment = &environment;
    %syslput project     = &project;
    %syslput protocol    = &protocol;
    %syslput startupfile = &startupfile;
    %syslput remote_yn   = &remote_yn;
    %syslput valid_pgm   = &valid_pgm;
    %syslput valid_lev   = &valid_lev;

    %*** Transfer control to CPI ***;
    rsubmit;
        %nrstr(%let root        =/opt/rootar;)
        %nrstr(%let protpath =&root/&environment/&project/&protocol;)
        %nrstr(%let fullstartupfile=&protpath./pgmsetup/&startupfile.;)
        %nrstr(%include "&fullstartupfile.";)

        %* Setting validation environment ;
        %if "&valid_pgm." ne "" %then %do;
            %if "&valid_lev." ne "" %then %do;
                %nrstr(%let valpath  = &protpath/validate/…;)
                %nrstr(libname   lptodv   "&valpath";)
                %nrstr(filename  fptog    "&valpath";)
            %end;
        %end;
    endrsubmit;
%mend startup0setup;
```

Figure 3. A sample macro code to implement remote submission.

The code present in Block A is essential for transferring the macro variable from the local environment to the server. The macro statement %syslput in the red box is to create a group of macro variables to the server session.

Code in block B is executed on remote server. The macro function in the red box ensures that the macro code is compiled and executed on the server.

To build this tool, two questions need to be addressed:

1. Submit and execute SAS code to remote server.
2. Transfer macro variables from local machine to remote server.

SUBMIT AND EXECUTE SAS CODE TO REMOTE SERVER

In SAS/CONNECT, statements inside the RSUBMIT block (Figure 3, Block B) are executed in the server session and all other statements (Figure 3, Block A) are executed in the local session. However, this behavior can be changed when you use a macro with an RSUBMIT statement to remotely submit code. As stated in SAS document: “When a macro is compiled, two results are produced: compiled macro statements and text. Even though they exist within the RSUBMIT block, these compiled macro statements, or instructional code, are executed in the local SAS session. Only the macro-generated text is passed to the remote server where it is executed remotely.”[1]

Here we need to understand 1) Code that is compiled runs on the local machine, and 2) Code that is read as text runs on the remote server. In our sample code shown in Figure 3, we used %NRSTR macro function to “hide” certain macro statements from the macro processor during compile-time. By hiding them, the specified statements are not compiled and executed locally by the macro processor. Instead, the SAS macro processor is instructed by the function to interpret the statement as text and send it to the remote session for execution.

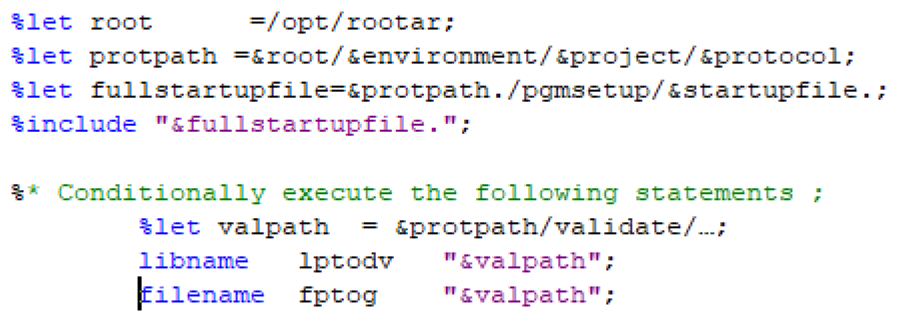
RSUBMIT block code in Figure 3 explanation:

1 The first four %nrstr(…) statements tell the macro processor these lines are text to send to remote server for execution.

2 The two %if statements are compiled and executed on local machine. These two statements help to set up appropriate working and output directory.

3 The three %nrstr(…) statements tell the macro processor these lines are text to send to remote server for execution.

If the two %if statements are true (Figure 3, Code block B), then the code execute on the server look like Figure 4.



```
%let root      =/opt/rootar;
%let protpath =&root/&environment/&project/&protocol;
%let fullstartupfile=&protpath./pgmsetup/&startupfile.;
%include "&fullstartupfile.";

%* Conditionally execute the following statements ;
        %let valpath  = &protpath/validate/…;
        libname   lptodv   "&valpath";
        filename  fptog    "&valpath";
```

Figure 4. The equivalent code executed on remote server If two %if statements are true in Figure 3 block B

TRANSFER MACRO VARIABLES FROM LOCAL MACHINE TO REMOTE SERVER

The %SYSLPUT and %SYSRPUT statements can help the macro variable is created on the local host and resolution tries to take place on the remote host or vice versa. By calling these functions, the variable defined in local can be carried to the server. In Figure 3, block A, we used %SYSLPUT statement to catch the macro parameters and pass the macros to the server. So, the RSUBMIT block is able to resolve the macro values.

CONCLUSION

This utility tool ensures that in each deliverable we maintain only one “core setup” file for programmers working on different platform and for different purposes such as develop/validation, on datasets or TFLs. It can greatly improve efficiency and accuracy and reduce frustration and avoid errors caused by inconsistent of setup parameters.

REFERENCES

1. SAS/CONNECT for SAS® Viya User’s Guide. “Using the Macro Facility with SAS/CONNECT”. https://go.documentation.sas.com/doc/en/vdmmlcdc/8.1/connref/n1wm9969gdtuijn195n1o7uaqp87.htm.

CONTACT INFORMATION

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

William Wei
Merck & Co.
William.wei@merck.com

Shunbing Zhao
Merck & Co.
Shunbing.zhao@merck.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Any brand and product names are trademarks of their respective companies.