Paper 383-2017

Setting Relative Server Paths in SAS® Enterprise Guide®

Michelle Buchecker, ThotWave Technologies LLC

ABSTRACT

Imagine if you will… a program, a program that loves its data, a program that loves its data to be in the same directory as the program itself. Together, in the same directory. True love. The program loves its data so much, it just refers to it by filename. No need to say what directory the data is in; it is the same directory.

Now imagine that program being thrust into the world of the grid. The grid knows not what directory this program resides. The grid is an uncaring, but powerful, soul. Yet, when the program is executing and the program refers to the data just by filename, the grid bellows “nay, no path, no data”.

A knight in shining armor emerges, in the form of a SAS Macro, who says “lo, with the help of the Enterprise Guide macro variable minions, I can gift you with the location of the program directory and send that with you to yon mighty grid.”

And there was much rejoicing. Yay.

This paper will show you a SAS macro that you can include in your Enterprise Guide pre-code to automatically set your present working directory to be the same as where your program is saved to on your Unix or Linux operating system. This then gives you the flexibility of moving your code and data to different locations without having to worry about modifying the code. It also helps you save time by not specifying complete path names in your programs. And can’t we all use a little more time?

INTRODUCTION

When submitting SAS code that is processed on a server, whether that server be a BI workspace server or a SAS Grid server, the code is passed along until it encounters that server. By the time the code hits the server, the server does not know the directory from which that code was originally stored.

This becomes a problem if your code uses relative paths for filenames or libraries. In many instances, you have code out there that uses a relative path, that is relative to where the code is stored. Or you may want to use a relative path of where the Enterprise Guide project is stored. This relative path is also sometimes known as the current working directory (cwd) or present working directory (pwd).

When your code gets to the executing server, it still has a cwd, but this is not the same one as where the code originated.

In this paper, you will see how to take the physical location where the code or Enterprise Guide project is stored and use it to create a macro variable. It is this macro variable value that is set as the cwd location.

MACRO VARIABLES NEEDED

First, let’s discuss the macro variables that need to come to our rescue and what they mean.

The three macro variables needed are &_CLIENTPROJECTPATH, &_SASSERVERNAME, and &_SASPROGRAMFILE.

- &_CLIENTPROJECTPATH contains the path and name of the saved Enterprise Guide project.
- &_SASSERVERNAME is used in this context to help us determine if the program is running locally or on a server.
- &_SASPROGRAMFILE contains the path and name of the saved SAS program. NOTE: For this macro variable to be populated effectively for code run on a server, you must be running EG 7.13 HotFix 3 which is described in SAS Usage Note 59744. Earlier releases only populated this macro variable if the saved program is opened using the Servers shortcut (IE: not through a

mapped drive, or a newly created program. Only by using File > Open > Program > Server would this macro variable be populated prior to the hotfix.).

SETTING THE CORRECT DIRECTORY FOR SAVED PROGRAMS

The most important part is determining where the desired current working directory is located. This is the true love location.

DETERMING WHERE THE PROGRAM IS SAVED

Using the &_SASPROGRAMFILE macro variable is an easy way to determine where the program should be saved. The downside is that this macro variable contains the path of the program and, the program name itself, and it is inside of quotation marks.

For instance, if I have opened the program through File > Open > Program > Server and my server is a Unix/Linux server, the value may be:

```
_SASPROGRAMFILE '/mbuchecker/programs/my_pgm.sas'
```

If I am working on a Windows server, the value may be:

```
_SASPROGRAMFILE 'C:\Users\mbuchecker\Documents\My SAS Files\my_pgm.sas'
```

SETTING THE CORRECT DIRECTORY LOCATION

Now that we know the value of the macro variable, we need to extract just the path. This entails:

- removing the quotes
- extracting all text except for the program name

Of course there are a myriad of ways to go about this. Here is one solution.

To remove the quotes, use the COMPRESS function, specifying &_SASPROGRAMFILE as the first argument and a single quote as the second argument. This will remove all single quotes from the text string. Because I want to use the COMPRESS function outside of a DATA step, I'll need to use either %SYSFUNC or %QSYSFUNC which allows me to use most DATA step functions outside of a DATA step, such as on a %LET statement. Because the value returned could have unusual characters that could wreak havoc with my program, I've opted for the safety of %QSYSFUNC:

```
%let noquotes=%qsysfunc(compress(&_SASPROGRAMFILE,"'"));
```

Next, extract all text except for the program name. Once again there are several options, but I use this two-step process:

1. Find only the name of the program.
2. Extract all characters up to where the program name starts as determined by the number of characters in the program name.

To do Step 1, I used a basic %QSCAN function using the value of the previous macro variable as the first argument, -1 as the second argument, and a forward slash as the third argument. In English this reads as: Scan &noquotes looking for the last word where words are only separated by forward slashes, and then quote or protect the results. Note that if you were to use this code for a Windows server, you will need to change the forward slash to a backslash.

Getting just the program name:

```
%let pgmname = %qscan(&noquotes, -1,/);
```

Step 2 is to take the original text string without the quotes, and grab all characters until where the program name starts. How do we find out where the program name starts? An easy approach is to find

out how many characters are in the program name with the %length function, and subtract that from the total length of the macro variable that has both the directory and program name in it. Of course we have to subtract off just one more character which is the slash that separates the last directory from the file name.

Getting just the directory name:

```
      %let cwd=%qsubstr(&noquotes, 1, %eval(%length(&noquotes) -
  %length(&pgmname) -1));
```

Lastly, we are going to use the X command to issue a cd (change directory) command to the operating system. The cd command is applicable for both Windows and Unix/Linux operating systems. Along with the cd command, we use the macro variable from step 2 to set the directory where the program is stored to be the current working directory. We also add a helpful note in the log for verification.

Submitting the cd command:

```
      x "cd &cwd";
      %put NOTE: Current working directory has been changed to: &cwd;
```

By issuing the **cd** command to the operating system, that directory is now considered to be the current working directory. So if I have a LIBNAME statement:

```
      libname test './data';
```

the period in the LIBNAME statement says to use the current working directory, in this case that is the location where my program is saved to, and then go down to a UNIX/Linux subdirectory within that location named **data**.

This allows for easier maintainability of your programs as you may have the code and data in one location during QA and move everything to a different location for production. By using this macro to set the relative path directory based on the program, you do not need to modify the code.

ADDITIONAL FLEXIBILITY

I modified the original above code to have some additional flexibility. When calling the macro, the user can choose to:

- Allow the location of the saved Enterprise Guide project to be the current working directory.
- Override the current working directory with a completely different directory.

Specifying a method other than the above, or specifying a combination that doesn't make sense will result in writing an appropriate message to the log and quitting the macro,

To provide this additional level of flexibility, I created a macro with two parameters. The first indicates the current working directory from where the program should obtain the value. A value of 1 for this parameter means the location of the program. A value of 2 means the location of the project. A value of 3 means override the current working directory with the value specified by the second parameter.

The macro definition starts with:

```
%macro set_cwd(method=1, cwd=) ;
   /* method 1 indicates to use the location where the program is saved to
set the CWD (this is the default)
      method 2 indicates to use the location where the EG Project is saved
to set the CWD
      method 3 indicates to override the CWD with the specified path */
```

This code needs to be run prior to executing a SAS program. This technique is often referred to as "pre-code". To make this macro pre-code click Tools > Options > SAS Programs > Insert custom SAS code before submitted code.Enterprise Guide administrators can push this out to all users.

OVERRIDE THE DEFAULT CURRENT DIRECTORY

If the user passes in a second parameter value,(cwd is not null), then I figure if they went to the effort to type in a second parameter value then they must want to override the defaults.

Next, knowing users like we do, I verify that they didn't also specify a method of 1 (use the program location), or 2 (use the project location). If they specified either of those, I print a note out to the log that even though they didn't use the correct method, I'm still using the cwd that they specified.

Regardless, as long as the cwd parameter has a value, we

1. Change to the directory using the cd command
2. Put a note to the log saying it has been changed
3. Use a %RETURN statement to get out of the macro since there is no point doing additional processing.

Code to override directory:

```
  %if %superq(cwd) ne  %then %do; /* Use the overriding location specified
to set the CWD */
    %if &method eq 1 or &method eq 2 %then %put NOTE: CWD specified and
method other than 3 specified, using CWD specified;
     x "cd &cwd";
     %put NOTE: Current working directory has been changed to: &cwd;
     %return; /* get out of macro */
  %end;
```

USE ENTERPRISE GUIDE PROJECT LOCATION AS CWD

Using the Enterprise Guide project location as the current working directory is more challenging than using the program location. Instead of &_SASPROGRAMNAME, the macro variable needed is &_CLIENTPROJECTPATH. Ok, that wasn't too hard.

The hard part comes from all the possible permutations of where projects can be saved and where the program is running. Table 1: Enterprise Guide Project Scenarios~~Enterprise Guide Project Scenarios~~:

Scenario	Outcome
Project saved on Windows drive, program executing locally and not on server	Everything is local so set cwd as appropriate
Project saved on Windows drive, program executing on server	Mismatch, don't set cwd, and terminate macro
Project saved on server, program running on server	Reassign project name as program name and reuse code based on program name
Project not saved	Mismatch, don't set cwd, and terminate macro

Table 1: Enterprise Guide Project Scenarios for Project Locations

To determine if the code has been submitted locally or to the server, we need to use the &_SASSERVERNAME macro variable. This variable will have a value of **'Local'** if submitting locally, and something else if submitting to the server. Note that the value **does** contain quotation marks as part of the value.

First check if the project is saved locally on Windows and then we can branch from there:

```
  %if &method = 2 %then %do; /* Use location where EG Project is saved to
set the CWD */
     %if %qsubstr(&_CLIENTPROJECTPATH, 3, 1) eq %str(:) %then %do; /* EG
project is saved to a Windows drive */
```

If this condition is true that means that the project is saved on Windows. We then need to check where the program is executing. If it is executing on the server, this is a conflict. Write a note to the log and terminate the macro:

```
         %if &_sasservername ne 'Local' %then %do;
             %put NOTE: Enterprise Guide project saved on local drive but
program executing on server. Current working directory not set;
             %put cwd is &cwd;
              %return; /* get out of macro */
         %end;
```

Next we address the %ELSE section. To recap, to get to this point means we have a locally saved project running locally. So we need similar code to what we did earlier when extracting the program path location, except this time we extract the project path location:

```
         %else %do; /* local EG project running on local host */
             %let noquotes=%qsysfunc(compress(&_CLIENTPROJECTPATH,"'"));
             %let pjtname = %qscan(&noquotes, -1,/);
             %let cwd=%qsubstr(&noquotes, 1, %eval(%length(&noquotes) -
%length(&pjtname) -1));
             x "cd &cwd";
             %put NOTE: Current working directory has been changed to: &cwd;
             %return;
         %end;  /* local EG project running on local host */
```

At this point we've exhausted instances where the project has been saved locally. The %RETURN statements have terminated the macro in this case. This means that we still need to address conditions where the EG project has been saved to the server, or not saved at all.

If the project has been saved, well that's no different than the logic above for of the code being saved. So I took a shortcut and just assigned the &_CLIENTPROJECTPATH macro variable value to &_SASPROGRAMFILE, and then let it fall thru to the rest of the macro code that sets the cwd based on &_SASPROGRAMFILE, IE: Method=1. (See above section for that code.)

Here is the snippet of code that tests &_CLIENTPROJECTPATH and sets &_SASPROGRAMFILE to &_CLIENTPROJECTPATH if the project was saved on the server:

```
     /* below code runs when EG Project is not saved locally. Could be
saved to a server or not saved at all */
     %if &_CLIENTPROJECTPATH ne %then %do; /* project saved */
       %let _SASPROGRAMFILE = &_CLIENTPROJECTPATH ; /* code then should
execute the method 1 code below */
     %end;
```

If the project hasn't been saved, &_CLIENTPROJECTPATH will have a null value. In this case, write a note to the log stating that a project name was not detected and terminate the macro.

```
     %else %do;
        %put NOTE: No project name detected. Current working directory not
set. Save the project to use this method.;
        %return; /* get out of macro */
     %end; /*project not saved */
```

```
  %end; /* end method = 2 */
```

COMPLETE CODE

Here is the entire program:

```
%macro set_cwd(method=1, cwd=) ;
options mlogic symbolgen;
   /* method 1 indicates to use the location where the program is saved to
set the CWD (this is the default)
      method 2 indicates to use the location where the EG Project is saved
to set the CWD
      method 3 indicates to override the CWD with the specified path */
  %if &method ne 1 and &method ne 2 and &method ne 3 %then %do;
     %put NOTE: A method of &method is not valid. Valid methods are 1, 2,
3. Current working directory not set.;
     %return; /* get out of macro */
  %end;

  %if %superq(cwd) ne  %then %do; /* Use the overriding location specified
to set the CWD */
    %if &method eq 1 or &method eq 2 %then %put NOTE: CWD specified and
method other than 3 specified, using CWD specified;
     x "cd &cwd";
     %put NOTE: Current working directory has been changed to: &cwd;
     %return; /* get out of macro */
  %end;

  %if &method = 2 %then %do; /* Use location where EG Project is saved to
set the CWD */
     %if %qsubstr(&_CLIENTPROJECTPATH, 3, 1) eq %str(:) %then %do; /* EG
project is saved to a Windows drive */
        %if &_sasservername ne 'Local' %then %do;
            %put NOTE: Enterprise Guide project saved on local drive but
program executing on server. Current working directory not set;
            %put cwd is &cwd;
             %return; /* get out of macro */
        %end;
        %else %do; /* local EG project running on local host */
            %let noquotes=%qsysfunc(compress(&_CLIENTPROJECTPATH,"'"));
            %let pjtname = %qscan(&noquotes, -1,/);
            %let cwd=%qsubstr(&noquotes, 1, %eval(%length(&noquotes) -
%length(&pjtname) -1));
            x "cd &cwd";
            %put NOTE: Current working directory has been changed to: &cwd;
            %return;
        %end;  /* local EG project running on local host */
     %end; /* EG project is saved to a Windows drive */

     /* below code runs when EG Project is not saved locally. Could be
saved to a server or not saved at all */
     %if &_CLIENTPROJECTPATH ne %then %do; /* project saved */
       %let _SASPROGRAMFILE = &_CLIENTPROJECTPATH ; /* code then should
execute the method 1 code below */
     %end;
     %else %do;
```

```
        %put NOTE: No project name detected. Current working directory not
set. Save the project to use this method.;
        %return; /* get out of macro */
     %end; /*project not saved */
  %end; /* end method = 2 */


/* method is 1,*/
   %if %superq(cwd) ne %then %do; /* using overriding directory */
      x "cd &cwd";
      %put NOTE: Current working directory has been changed to: &cwd;
      %return;
   %end;

  %if &_SASPROGRAMFILE ne %then %do;   /* Use location where program is
saved to set the CWD */
    %let noquotes=%qsysfunc(compress(&_SASPROGRAMFILE,"'"));
    %let pgmname = %qscan(&noquotes, -1,/);
    %let cwd=%qsubstr(&noquotes, 1, %eval(%length(&noquotes) -
%length(&pgmname) -1));
    x "cd &cwd";
    %put NOTE: Current working directory has been changed to: &cwd;
  %end;
  %else %do;
    %put NOTE: No program name detected. Current working directory not set.
Save the program or open the program from the server.;
  %end;
%mend;
```

CONCLUSION

Using this robust macro as pre-code in your Enterprise Guide project allows you to use relative paths within your program: based on where the program is saved, where the project is saved, or an overriding directory. This gives you the flexibility of moving your code and data to different locations without having to worry about modifying the code. It also helps you save time by not specifying complete path names in your programs. You can now keep your program and your data married together in the same directory. And even if these lovers move you know that the server will still recognize them as a couple.

ACKNOWLEDGMENTS

The ThotWave Technologies team for support and suggestions.

CONTACT INFORMATION

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

Michelle Buchecker
ThotWave Technologies, LLC.
mbuchecker@thotwave.com