



SETTING THE ENVIRONMENT WITH SAS DM STATEMENTS

Megha Agarwal, Clinovo, Sunnyvale CA

WUSS 2011

Annual Conference October 2011



TABLE OF CONTENTS

1. ABSTRACT	3
2. INTRODUCTION	3
3. SYNTAX	4
4. CONCLUSION	8

1. ABSTRACT

SAS programmers usually lose a lot of time (and temper) when setting up a correct environment before submitting a program. For example, every time you run a program, you need to close all the datasets that are created in the program, clear the log and output windows, save the program etc. Then once the program is run, you need to open a specific dataset to check the results. In a nutshell, you always follow some fixed, time-consuming processes, before and after executing the program.

Before knowing about the DM (Display Manager), I used to say, "if only all this could be done automatically". Then DM statements came to my rescue. A DM statement stands for Display Manager Statements. It submits SAS Program Editor, Log, Procedure Output or text editor commands as SAS statements. This paper will explain the purpose, functionalities and usage of DM statements using several examples.

The intended audience is all levels of SAS users.

2. INTRODUCTION

While coding and executing the code, many redundant tasks are to be done manually. DM statements really prove to be a boon to avoid such redundancies. To date, there is no proper reference paper or document that highlights the power of DM statements. This paper aims to familiarize you with the intricacies of DM. It includes commands for you to play and automatize your chores. Happy Coding!!

3. SYNTAX

DM <window> 'command(s)' <window> <CONTINUE>;

window	Specifies the active window. (like wedit, log, output)
command	Can be any windowing command or text editor command and must be enclosed in single quotation marks. If you want to issue several commands, separate them with semicolons. (like wedit, log, output). (like clear, wpaste etc)
CONTINUE	Causes SAS to execute any SAS statements that follow the DM statement in the Program Editor window and, if a windowing command in the DM statement called a window, makes that window active.

The default is the editor window through which you submit the code. This example would clear the concept of the term <window> and <CONTINUE> as used in the syntax. Let's understand it with the help of the log clear statement. For example:

If you want to clear the log and return back to the editor window after submitting DM statement then:
Dm log 'clear;';

Else if you want to remain at the log window then

Dm log 'clear;' continue;

If you want to clear the log and output both, and then return to log window, then

Dm 'log; clear; output; clear;' log;

1) Now, each time you run or debug the code, you have to save the code, clear the log and output, close the open datasets, run them, then search the library for the dataset you are working and open it. Imagine the time and clicks you save using the following code:

```
dm 'flsvlast';                *saves the program;
dm 'log; clear; output; clear;'; *clears log and output;
dm 'next VIEWTABLE;; end;';   *closes the open dataset;

*your code. In this case test creation of dataset.
```

2) If the name of the dataset created is dynamic then you can use the following, which opens the last created dataset.

```
dm "vt &syslast";
```

3) Usually when you are working with a dataset, you want to see the column names and not the column labels. The following statement helps view the column names for a specific dataset, but for other datasets, column labels would be visible.

```
dm "VT libname.dataset COLHEADING=NAMES" continue;
```

4) If you are adding a column to the dataset, or changing the attributes of a dataset, then you are not really interested in the entire data but just the columns and its attributes. The following command will open the column attribute window of the desired table, and not the table.

```
dm 'var libname.dataset;' continue;
```

5) To clear a cluttered results window:

```
dm 'odsresults' clear ;
```

6) If you are paranoid about using proc export and import, and you just want to do the basic export/import, then the following DM statement is for you. It works with excel and txt files.

```
*will create multiple sheets if same excel is referred multiple times  
with different datasets;  
  
dm "DEXPORT libref.dsn 'filename.xls' replace";  
  
*will import the last sheet in case of excel and the entire file in  
case of txt:
```

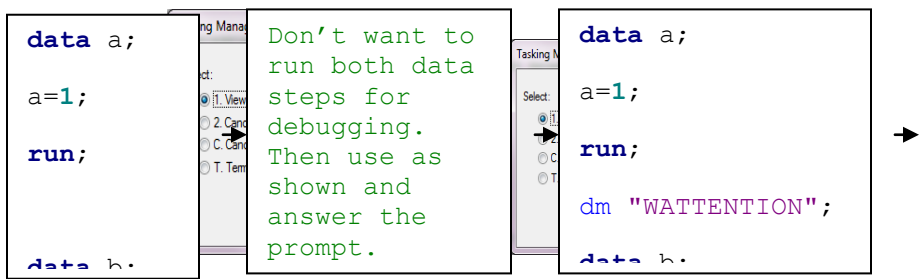
7) If you are assigning a libref or fileref, then you are always wary of which file or library reference names are already occupied. If you use the same reference for multiple files, it could damage the whole data. The shortcut to check them is:

```
dm 'filename'; *shows all the filenames;
dm "libname". *shows all the libnames;
```

8) You usually use proc printto to route our log/output to a specific location. The drawback of doing so, is that you can no longer see them in your log and output windows. Following command would be a great resort:

```
dm log 'file "filepath.filename.extension" '; *saves log in the
specified file;
dm out 'file "filepath.filename.extension" ';*saves output in the
```

9) At times, in interactive mode, you wish to stop your program from going in an infinite loop, or during debugging you don't want to run the whole code but a selection, (but don't want to select each time you run). You can use the following, which will give you a prompt, asking to cancel submitted the code..



10) If you want to close any window programmatically, then:

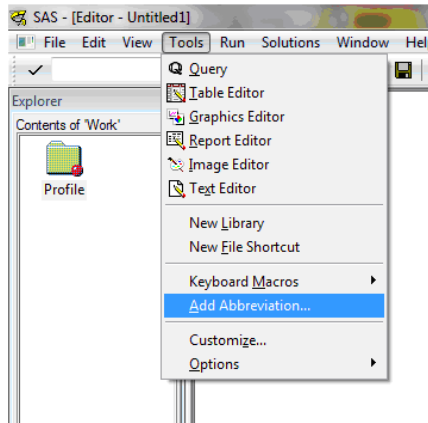
```
dm log/out/wedit 'winclose';*closes the specified window;
```

11) By combining DM commands, you can copy from and paste in a file. For example, if you want to add a header, or some constant code/text, or copy from one file and paste in other file, the following code can do it without any user intervention:

```
*this brings the text to be copied to the clipboard;
dm 'whostedit; include "filepath.filename.extension";
EDCMD selectall; EDCMD copy; EDCMD winclose; ';

*this puts it at the beginning of each of the required files;
```

12) Last but not the least, by a single click, you can insert uniform blocks of text into your program with a minimum of typing, using display manager’s “Abbreviation” and edit this abbreviation or record new macros using “Keyboard Macros”. While running the Display Management System from within SAS, select the pull down menus Tools and Add Abbreviation. This will open a dialog box. Enter the text/code/comments, and save it. Whenever you want to use an abbreviation, simply type the name of the abbreviation in the Enhanced Editor. As soon as the last letter of the abbreviation has been entered, a small pop-up ‘tip’ text box containing the first few characters of the abbreviation is displayed. If at that point, the Tab/Enter key is pressed, then the name of the abbreviation will be replaced by the text that you stored.





4. CONCLUSION

SAS has unlimited power in many of its unusually used statements. Rather than using the conventional methods, unraveling the mystery behind those statements could unleash newer and more expedient coding standards.

CONTACT INFORMATION

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

Name: Megha Agarwal

Enterprise: Clinovo

Address: 1208 East Arques Avenue

City, State ZIP: Sunnyvale CA 94085

Work Phone: 408-773-6251

E-mail: megha.agarwal@clinovo.com

Web: www.clinovo.com <http://blog.clinovo.com/> http://tech.groups.yahoo.com/group/sas_academy/

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