Paper 9000-2016

Kicking and Screaming Your Way to SAS® Enterprise Guide®

M. Michelle Buchecker, ThotWave Technologies, LLC.

ABSTRACT

You are a skilled SAS® programmer. You can code circles around those "newbies" who point and click in SAS® Enterprise Guide®. And yet… there are tasks you struggle with on a regular basis, such as "Is the name of that data set DRUG or DRUGS?" and "What intern wrote this code? It's not formatted well at all and is hard to read." In this presentation you learn how to program, yes program, more efficiently. You learn the benefits of autocomplete and inline help, as well as how to easily format the code that intern wrote that you inherited. In addition, you learn how to create a process flow of a program to identify any dead ends, i.e., data sets that get created but are not used in that program.

INTRODUCTION

So why use Enterprise Guide? First let me give you a little history of Enterprise Guide. And before that, let me give you a little history of SAS. SAS was incorporated July 1, 1976 by 4 professors from a NC university as a programming language. Now if we think back to the 70's only programmers were using computers so all was good. As time progressed non-programmers started using computers and needed the functionality of what SAS did. But since they weren't programmers by nature they said "SAS is too hard".

So SAS Institute created Enterprise Guide, or EG, in 1999 for these users to point and click their way thru writing programs. And what they found out was not only were non-programmers using EG, but programmers were as well.

So SAS Institute asked them "Why are you using EG? We didn't build it for you since you know how to code."

The programmers replied "yes, but we don't know how to code EVERYTHING, like PROC GCHART especially, because that's really hard. So we use EG to write most of the code for us that we don't know and then tweak the results".

And SAS said "oh, we didn't think of that".

Then SAS went about making EG a better programming interface so that programmers could write the code that they knew in EG, plus have the benefit of pointing and clicking thru code they are less familiar with all in 1 interface. As such, EG ended up to be a better programming editor than Display Manager and we'll see some of those extra benefits in this paper.

In this paper you will learn the basics of Enterprise Guide, such as how to write, submit, and debug programs. And you'll see the advantages of doing these common tasks over the more traditional ways. By using Enterprise Guide to write, submit, and debug your programmers you will perform these tasks faster and as a result you will be a more efficient SAS programmer.

In addition, you'll see a few statements that you need to avoid in Enterprise Guide.

BASICS OF USING ENTERPRISE GUIDE

Enterprise Guide is a programming interface. Yes, programming interface. Just like other SAS programming interfaces, you need a place to write the code, a way to submit the code, the ability to view the log, and the ability to view the output.

CREATING, OPENING, AND SUBMITTING PROGRAMS

To build a SAS program, select **File** ⇨ **New** ⇨ **Program** to create a new code node in the project.

To add a shortcut in the project to an existing SAS program, select **File** ⇨ **Open** ⇨ **Program…**, and navigate to where the program is stored. It could be stored locally on your machine, a mapped network drive, or if using SAS Business Intelligence or SAS Grid you may choose to open the program from the server.

A SAS program can be submitted using one of these techniques:

- Select **Run** or **Run Selection** from the toolbar.
- Right-click on the program and select **Run** or **Run Selection**.
- Press F8 or F3.

When using these techniques it will submit the entire program. If you are running EG locally, all the SAS processing will occur on your local machine. If you are connected to a metadata server, for example if using SAS Business Intelligence, or the SAS Grid, then the code will be sent to the server for processing.

If you want to submit just a portion of your program, highlight the code you wish to execute and use one of the submission techniques listed above.

VIEWING THE LOG AND LOG SUMMARY

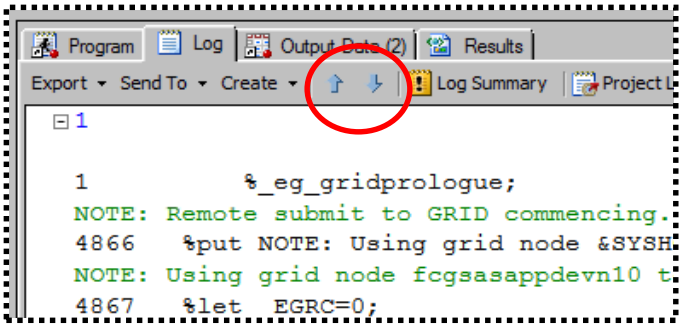
After you submit the program, you will either be taken to the Log tab or the Results tab. If your code did not request any reports, then you are taken to the Log tab. If your code requested reports, but there were errors in the program, you are taken to the log tab. Only if your code generated a report **and** there were no errors would you be taken to the Results tab when the code has finished running.

To navigate manually to the log, it is accessible on the Log tab. The code icons in the project indicate whether there are warnings or errors in the SAS log.

Display 1. Icons of Programs in Process Flow

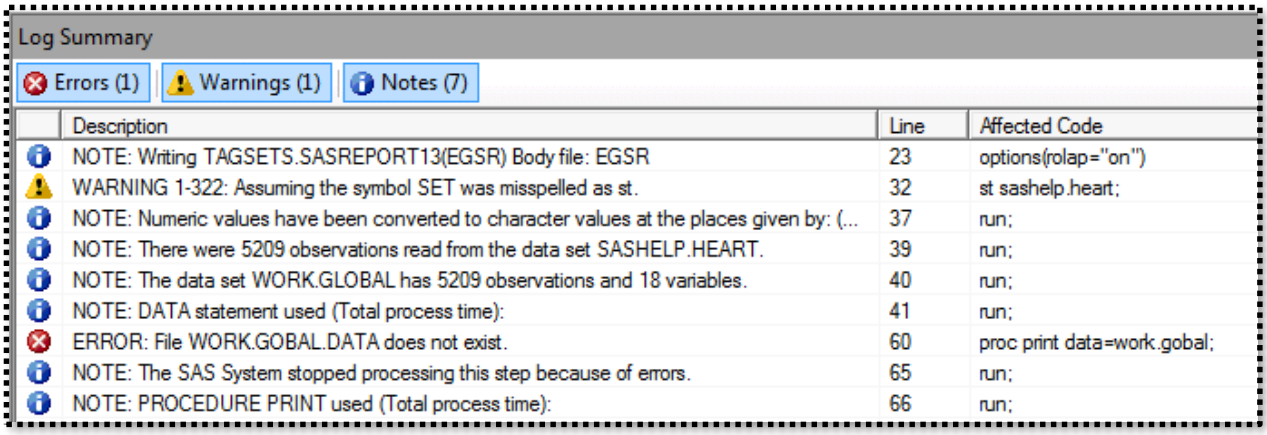
As every good SAS programmer knows you should always, always, always check the log for error messages, warning messages, and notes that look unusual.

Arrows on the Log tab enable quick navigation to the next warning or error.



Display 2. Search Log for Next and Previous Warning or Error

The Log Summary window can be turned on to show Notes, Warnings, and/or Errors. The Log Summary window lists how many errors, how many warnings, and how many notes are in the log.



Display 3. Log Summary

The Errors, Warnings, and Notes tabs are all toggles. If you click on the Notes tab that will hide the notes so that you can concentrate on errors and warnings. Clicking the Notes tab again will unhide the notes.

Double-clicking on an individual Note, Warning, or Error message will take you to that part of the log where that message occurred. No more fruitless scrolling or searching. You can more easily and quickly identify the problems in your program. It is still up to you to determine why the problem occurred, but that's why you have the SAS programming skills you do.

BEING A MORE EFFICIENT PROGRAMMER

You may have your favorite editor, whether it be Display Manager (aka PC SAS) or vi on Linux, or some other editor. But unless you are "Rain Man", when you are writing code you have to stop, think "what is the name of that variable?" or "what is the syntax of the PROC UNIVARIATE statement?" or some other coding syntax. It can be time consuming to look this up.

You may have inherited code from a co-worker and not be familiar with a certain option they used, such as the BODYTITLE option on an ODS statement. Once again it can be time consuming to look up what this does.

And we've all encountered poorly formatted code. It could be code written by an intern, or code we wrote ourselves to be ad-hoc that has gained usefulness and is now being used in production. Taking the time in a traditional editor to format it for readability is time consuming.

Speaking of inherited code, have you ever had a program that was so long and convoluted that you didn't know what all the intermediate input and output data sets were? Not a big deal until someone says "we are making a change to the LABS data set, what impact does that have on your program?". Once again, it's time consuming to trace through what step the LABS program is being used, does that data then create a differently named data set that goes to another step that creates a differently named data set and so on?

Now you may not have given any of the above much thought on how this can be improved, but it can be. That's the beauty of using Enterprise Guide. In this section you will see how to use Enterprise Guide as a great programming interface that makes you a better and more efficient programmer.

USING AUTOCOMPLETE

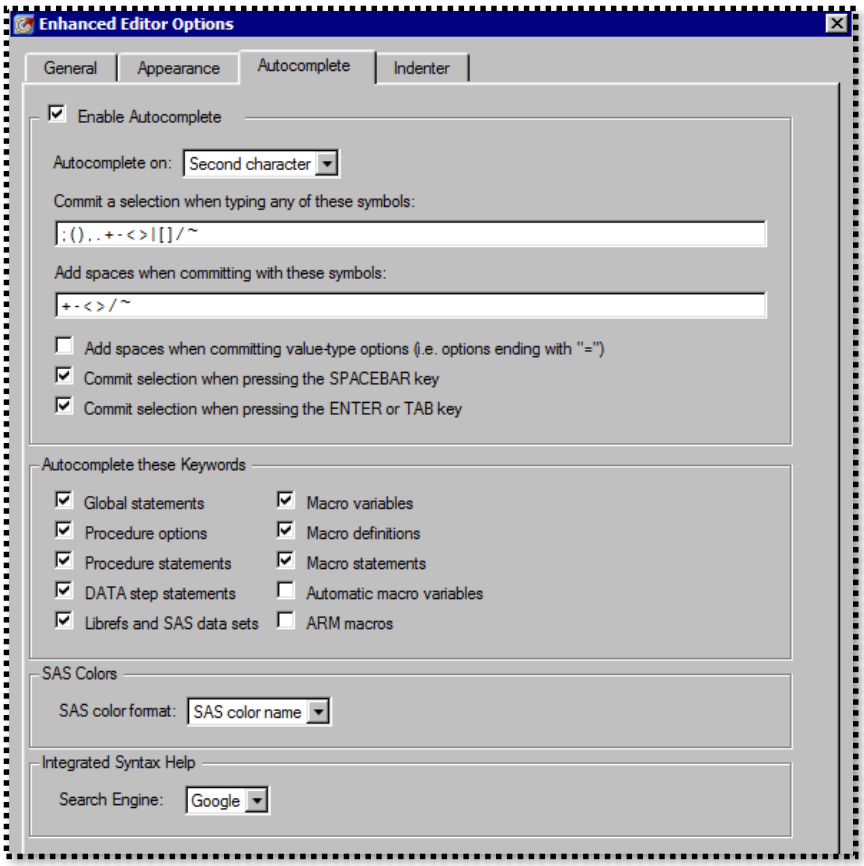
Think about the last time you couldn't recall the syntax on a program you were writing. Your typical next step is to go to your browser window and Google what you need. Depending on your Google-fu skills you may get the results immediately. But sometimes it takes multiple searches to specifically get what you want.

And if you search for something like **sas var statement univariate** the first hit is useful. Yay! But for some reason, Google often returns the 9.2 documentation links and not the current version. If you realize this, you can then add 9.4 to the search, but you may not even realize you are looking at outdated doc. All of this searching can be time consuming and is fraught with possibly getting outdated material.

Enterprise Guide solves all of this by adding the autocomplete feature to the SAS editor. The editor can suggest

- SAS statements
- procedures
- macro programs
- macro variables
- functions
- formats
- librefs
- SAS data set names
- SAS variable names

The Program Editor autocomplete options can be customized by selecting **Program** ⇨ **Editor Options**. Autocomplete can be customized or disabled on the Autocomplete tab.

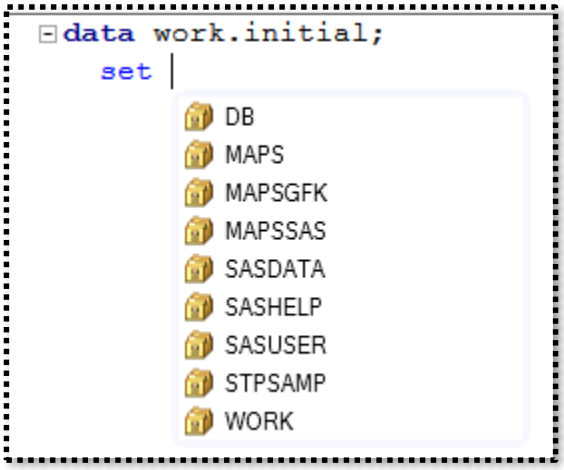


Display 4. Autocomplete Options

Notice that there are options to control how many characters you type before autocomplete kicks in, as well as what keys commit a selection. You can modify these options, or disable autocomplete altogether. In addition, you can control which keyword types get shown for autocomplete. So if you want to disable showing DATA step statements only, you could.

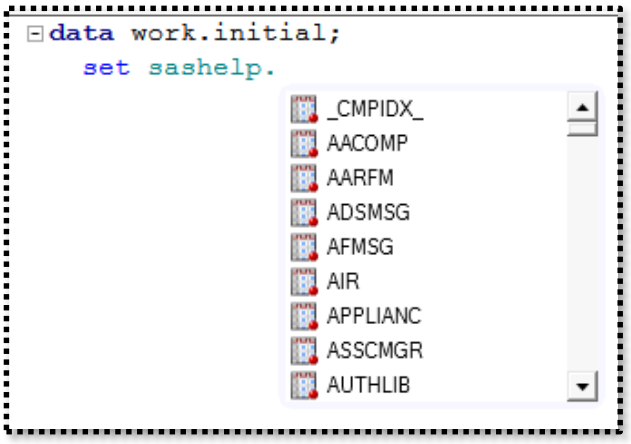
Not only is autocomplete useful for keywords, options, and statements, but it is also extremely useful for library names, data set names, variable names, macro names, and macro variable names.

For example, assume you are writing a SET statement in a DATA step. Using autocomplete, Enterprise Guide will list the names of the known libraries. You can then double-click on the name or type the first set of unique characters and press Enter.



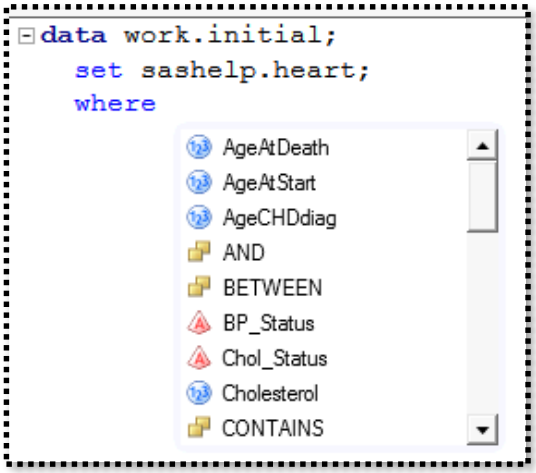
Display 5. Autocomplete LIBREF

Since SAS knows that the syntax is **libref.dataset** after you type a period after the libref, EG will then show you a list of data sets in that library. No more wondering if you named the data set **mylab**, **my_lab**, **my_labs**, etc. This saves you time from having to open up the library to find the exact name of the data set.



Display 6. Autocomplete Data Set Name

Using autocomplete on your own data is especially useful when you are specifying a variable name, such as on a WHERE statement. When a variable name is typically included in the code, Enterprise Guide will display the list of variables in that data set that was specified earlier in that step. How much of your programming career has been spent running PROC CONTENTS or opening up a data set just to find the name of a variable for your program? And if you are like me, sometimes I have to do it for the same variable multiple times if it has a cryptic name like LCHGBLXT.



Display 7. Autocomplete Variable Name

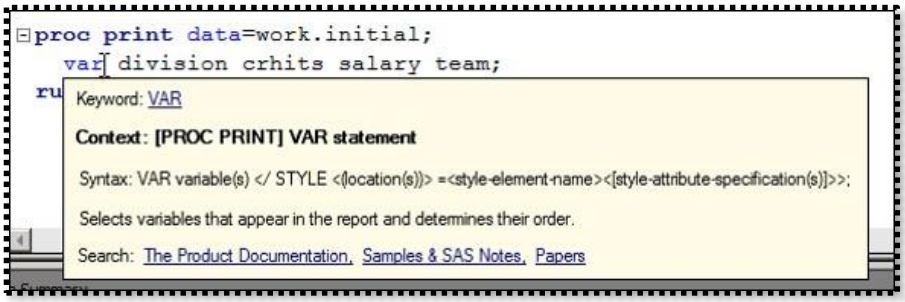
Notice that not only does Enterprise Guide display the name of the variable, but also the type. The blue circle with the **123** inside indicates that is a numeric variable. The red triangle with the **A** inside indicates it is a character variable. This is also useful information when you are coding so you know the correct operators to use for that type.

USING INLINE HELP

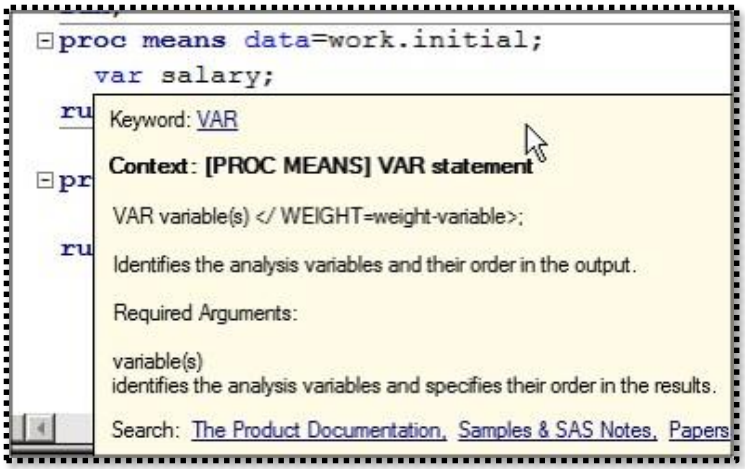
Periodically in our programming careers we have inherited code that we don't understand. If we weren't in EG, we would have to look up the documentation, or just ignore the code we don't understand (admit it, we've all done it). However, with the context sensitive in-line help, you can learn this code easily.

By hovering over a keyword, you will see the basic syntax for that keyword as well as the description. The inline help also include links to product documentation, samples and SAS Notes, as well as relevant papers.

In addition, the inline help is context sensitive. So if you hover over the VAR statement in a PROC PRINT step, the help is different than the VAR statement in a PROC MEANS step. Having that context sensitive help saves countless searches trying to get just the exact information you want online.

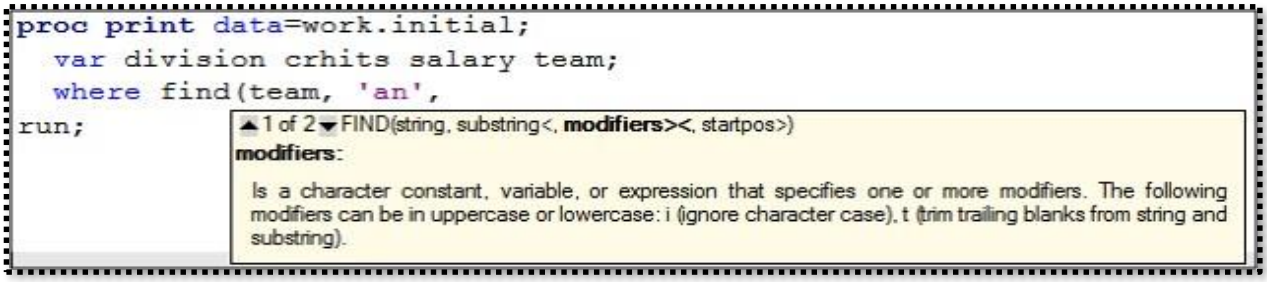


Display 8. Inline Help for VAR statement in PROC PRINT



Display 9. Inline Help for VAR statement in PROC MEANS

Other useful uses for inline help includes help on the arguments for functions. For example, the FIND function in SAS is a useful but not commonly used function and you may not always remember the arguments for it off the top of your head. When using the inline help, as you start typing the function arguments, you will see a pop up box indicating what argument you are on (listed in **bold**), and what is needed for that argument.



Display 10. Inline Help for FIND Function

Once again, this is a big time saver to have all the documentation you need at your fingertips for "one stop shopping" to write your code.

FORMATTING PROGRAMS

Picture this: you have inherited code from an intern. The code works fine, but the formatting of the code is hard to read. Or alternatively, you wrote some code for an adhoc purpose and didn't worry about making it pretty. And then it becomes useful code that you need to put into production.

Your inherited code may look like:

```
proc sort data=db.subjinfo
out=subjinfo; by usubjid;
run;

proc sort data = db.labs out = labs;
by usubjid visid;
where compress(lowcase(lbrestp)) = "cn";
```

```
data sub_labs;
merge subjinfo(in=a) labs(in=b);
by usubjid;
if a and b;
if lbepflgxt=1 then flag=
'flag was 1';
run;
```

Notice how there is no indentation, the BY statement on the first PROC SORT is on the same line as the OUT= option, and that the assignment statement on the IF/THEN is split onto multiple lines. To use a technical term, icky! And it is really time consuming to properly indent and fix the formatting issues in an editor that is not Enterprise Guide.

So how do you fix this using EG? Simple. From the pull down menus click **Edit** ⇨ **Format Code** or press CTRL+I on the keyboard. And voila!

The newly formatted code:

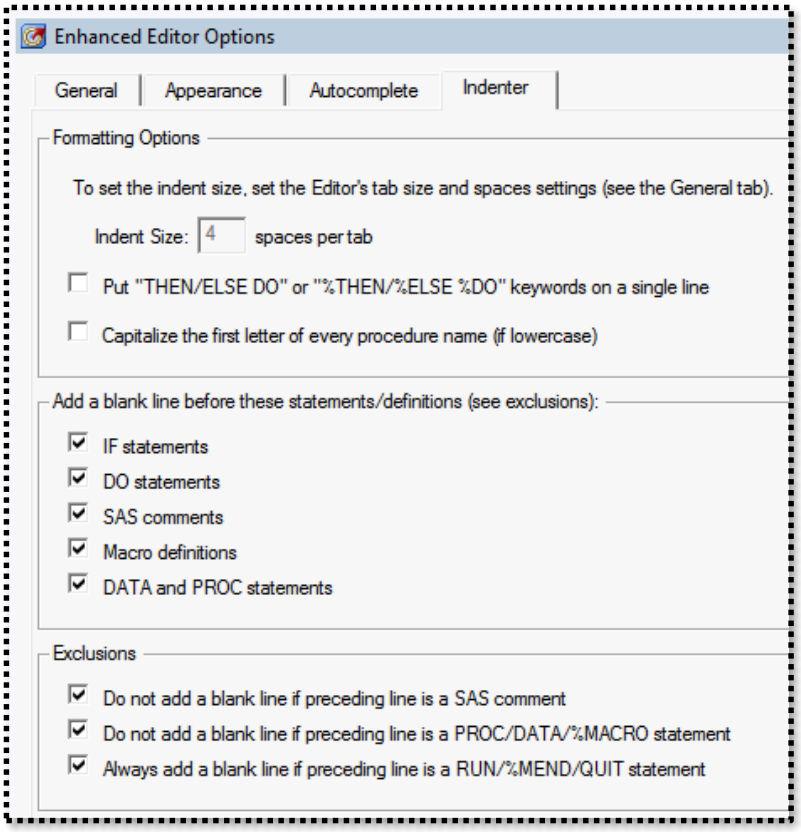
```
proc sort data=db.subjinfo
      out=subjinfo;
      by usubjid;
run;

proc sort data = db.labs out = labs;
      by usubjid visid;
      where compress(lowcase(lbrestp)) = "cn";

data sub_labs;
      merge subjinfo(in=a) labs(in=b);
      by usubjid;

      if a and b;

      if lbepflgxt=1 then
            flag= 'flag was 1';
run;
```

Notice that

- the formatted code is indented four spaces for each statement in a step
- the BY statement is on a separate line
- the IF/THEN statement is: IF condition THEN on one line and the action on the next line and indented
- there are blank lines between steps
- there are blank lines around each IF statement.

Now, I'm not a fan of four spaces for indentation. I prefer two. Some companies standardize on three. This is an easy fix. Click on **Program** ⇨ **Editor Options**. Then on the General tab, change the tab size from 4 to the number of spaces that you prefer.

You can also customize the insertion of the blank lines by selecting the Indenter tab and turning off the appropriate checkboxes.



Display 11. Indenter Options for Formatting Programs

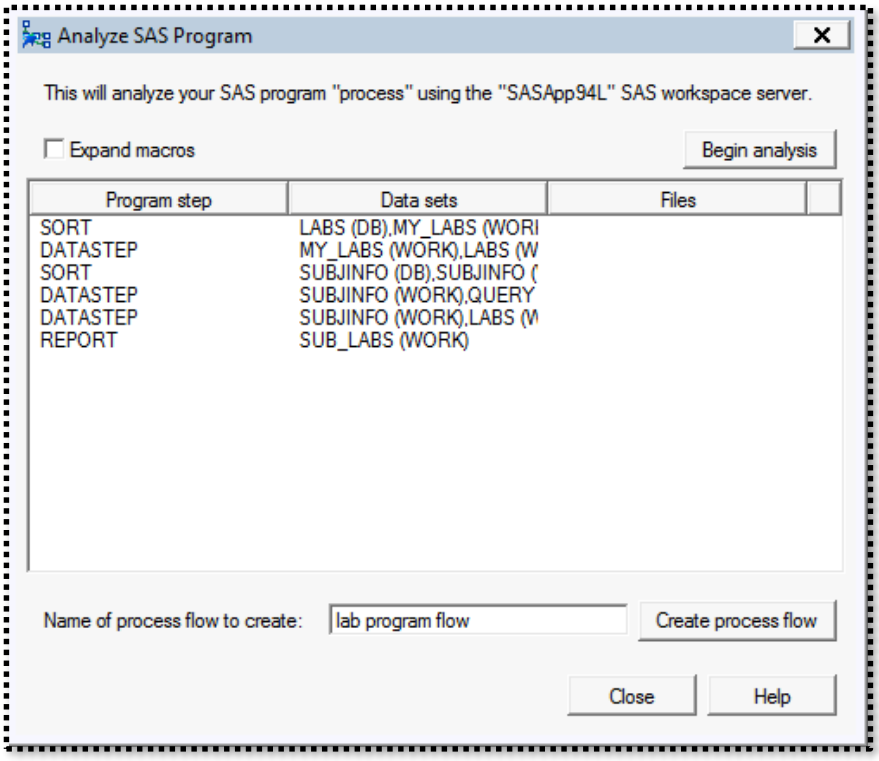
Easy peasy to easily format readable code.

CREATING A PROCESS FLOW FROM A PROGRAM

One of my major stumbling blocks as a programmer was that often the departmental code just grew and grew as new business changes were implemented. As the program grows, it becomes harder and harder to know what code can be safely deleted. Often we would leave in code that probably could be removed, but the program was so long and complex, we were unsure if a data set created was being used elsewhere in the program. Having unnecessary code in a program leads to longer processing time, as well as longer development time as you are trying to shoehorn additional code into the program.

Enterprise Guide has a fantastic approach that runs through the program and identifies all of the inputs and outputs for each step. This technique lets you document what the inputs and outputs are and you can identify data sets that get created that are never used. Once you have identified those data sets, you can then remove the code that created those data sets, which will speed up processing time as well as make the program easier to maintain.

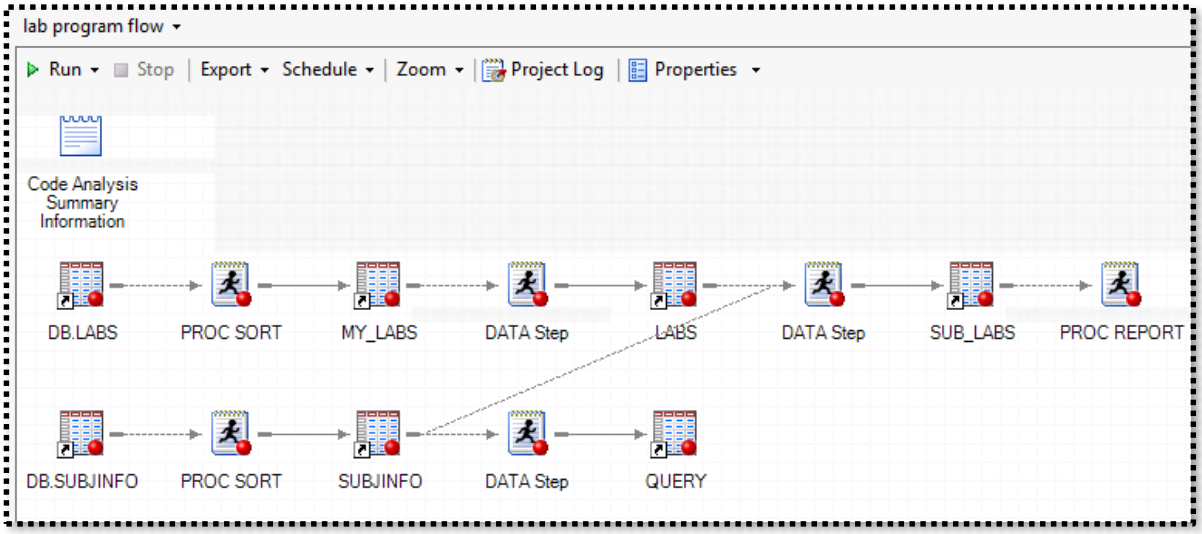
To analyze a program in Enterprise Guide, click **Analyze Program** ⇨ **Analyze for Program Flow**. Then click **Begin analysis**.



Display 12. Analyze SAS Program Window

Enterprise Guide will run the program and then provide a list of all steps and the data sets used in those steps. To create a visual representation of the program, provide a name for the process flow and click **Create process flow**.

A new process flow is created in the project depicting a visual representation of the program.



Display 13. Visual Representation of a Program after Analysis

In this visual representation, you can easily identify the inputs and outputs to each step. Here we can see that there is a data set named QUERY that is not being used in any subsequent step. This is a likely candidate for deleting the DATA step that creates this data set.

STATEMENTS TO AVOID

There are a handful of statements that you need to be aware of that work differently, or not at all, in Enterprise Guide.

They include:

- Code that would normally cause a window or prompt to appear (DEBUG, PROC FSLIST, DM) does not work and displays an error message in the log when used in SAS Enterprise Guide.
- Code that terminates the SAS process with ABORT or ENDSAS calls terminates the connection between SAS Enterprise Guide and the SAS server if you are not using a local version of SAS.

CONCLUSION

Using SAS Enterprise Guide provides a great interactive development environment to help you program more quickly. In this paper you learned the basics of using Enterprise Guide, specifically how to

- create, open, and submit new SAS programs
- view the Log and Log Summary

In addition, you learned techniques to being a more efficient programmer. Namely, how to

- use the autocomplete feature
- use inline help
- format programs for readability
- create a process flow from a program to identify inputs and outputs

And lastly, you can list programming statements to avoid in Enterprise Guide.

You may have been kicking in tantrum, and screaming in pain before you started using EG, but now you should be kicking your heels up and screaming with joy with all of the ways Enterprise Guide can improve your programming efficiency.

ACKNOWLEDGMENTS

The ThotWave Technologies team for support, suggestions, and editing. Stacey Syphus at SAS Institute for teaching me Enterprise Guide with her contagious enthusiasm.

RECOMMENDED READING

- *The SAS® Dummy Blog*

CONTACT INFORMATION

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

Michelle Buchecker
ThotWave Technologies, LLC.
mbuchecker@thotwave.com
http://thotwave.com/