Paper 070-2009

# SCL Is Gone: How Do I Get Variables from My Users into SAS® Enterprise Guide®?

Patricia Hettinger, IBS, Oakbrook, IL

## ABSTRACT

**SCL (SCREEN CONTROL LANGUAGE) HAS BEEN DISCONTINUED WITH SAS® ENTERPRISE GUIDE®. THE PURPOSE OF THIS PAPER IS TO ACQUAINT USERS WITH ITS REPLACEMENT, PARAMETERS MANAGER. THE PAPER ALSO GIVES SOME TIPS FOR DEALING WITH SITUATIONS WHERE IT FALLS SHORT, SUCH AS IN MASKING DATABASE USER IDS AND PASSWORDS. MANY SCREEN SHOTS ARE INCLUDED AS SAS ENTERPRISE GUIDE IS A VISUAL INTERFACE.**

## INTRODUCTION

SAS® Enterprise Guide® is the GUI interface beginning with SAS® 9. The GUI concept may take some getting used to for those of you who have been coding SAS® for a long time. How do you get input from your users if Screen Control Language (SCL) is no longer available? Parameters Manger replaces SCL as the input screen method for obtaining values for your macro variables. Although not a complete package, Parameters Manager can be used in some fairly sophisticated ways. This paper discusses its uses as well as some related SAS® Enterprise Guide® features.

## CREATING PARAMETERS (MACRO VARIABLES) FOR USE IN YOUR PROJECT

First set up parameters by selecting 'Parameters (Macro Variables) Manager from the Tools menu as demonstrated in Figure 1:

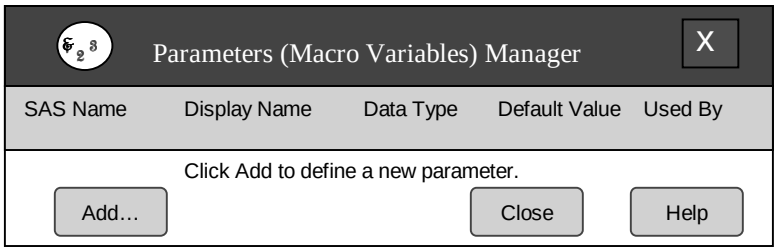



Figure 1: Initial Parameter Manager Screen

Click 'Add'. For our first parameter, you will create a string variable to be used as a database login id. The value in the 'Display Name' field is what will show on the screen when the parameter is prompted (Log in as user id: ). The SAS® code name will be userid. You will describe it as 'user id for database login'. The data type will be string as in Figure 2:

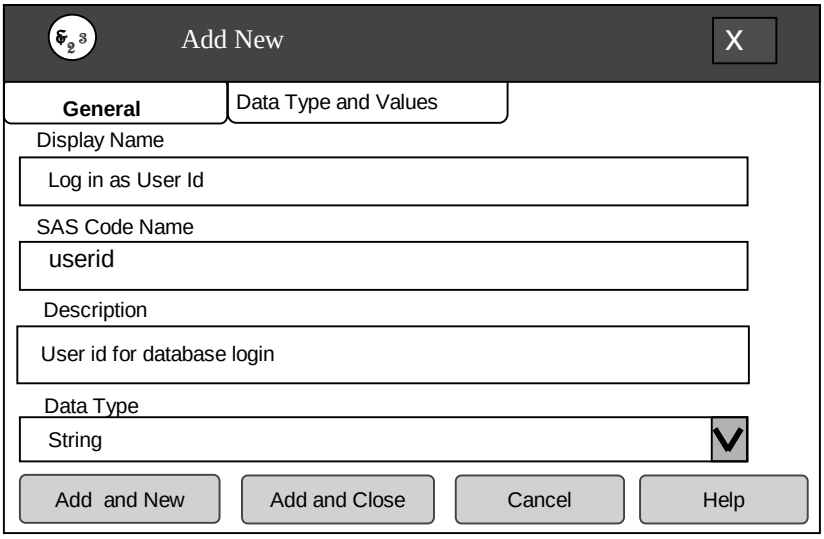



Figure 2: Add General Screen

Some work has to be done on this variable before it can be used, however. Click on the 'Data Type and Values' tab. You could provide some allowable values and even a default. But since it's a user id, skip that and set some more relevant properties (Figure 3) like requiring user input at runtime, prompting for the value and masking user input.

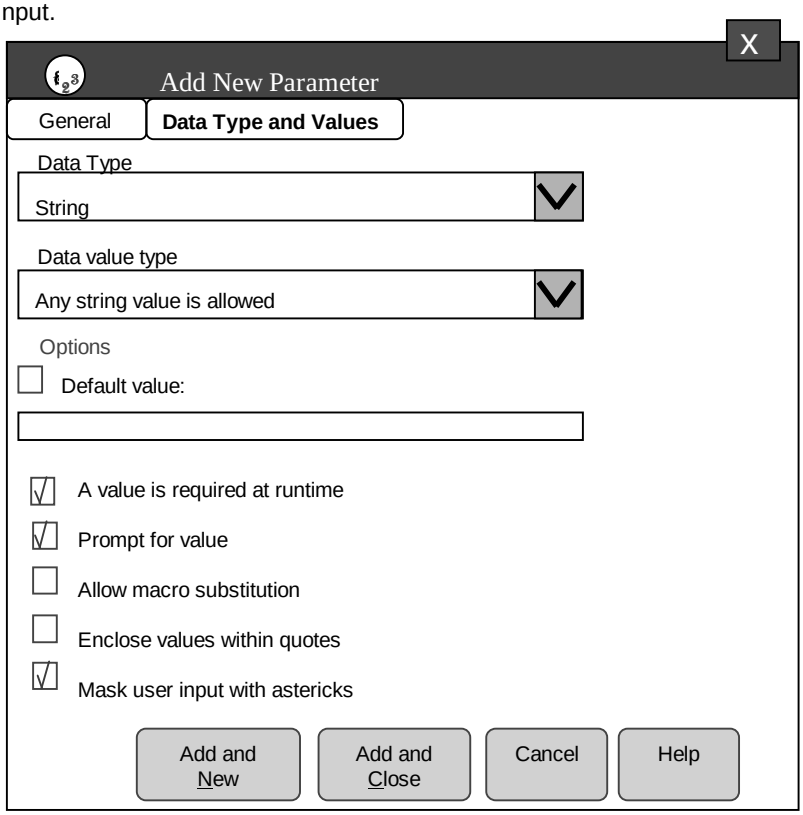



Figure 3: Variable properties

Click 'Add and New' (Figure 3) because you need to prompt for a password as well. Set the properties to be the same as the user id. Clicking the "Add and Close" button gives us information about both variables (Figure 4):

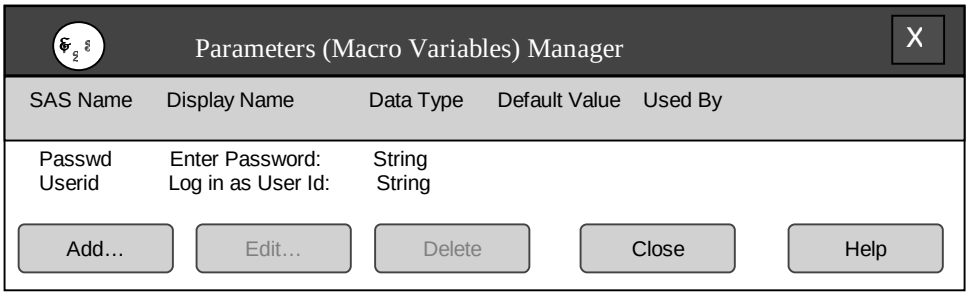



Figure 4: Variable List

## USING PARAMETERS

Now that you have created the parameters, you need to be able to use them when needed. For example, if we were connecting to a database, our code might look like this:

```
Proc sql;
Connect to database connection (database=proddb tdpid=prodpid user=&userid
password=&passwd.);
Create table extract1 as
Select * from connection to database connection (select statement here).
```

This SAS® code would have to be accessible to your project, either by embedding it in the project or linking to an external program. Here we will call it 'DB Login'. Wherever it's located, the procedure to connect your code to your parameters is the same. Right click on the code icon, and select 'Properties…' (Figures 5):

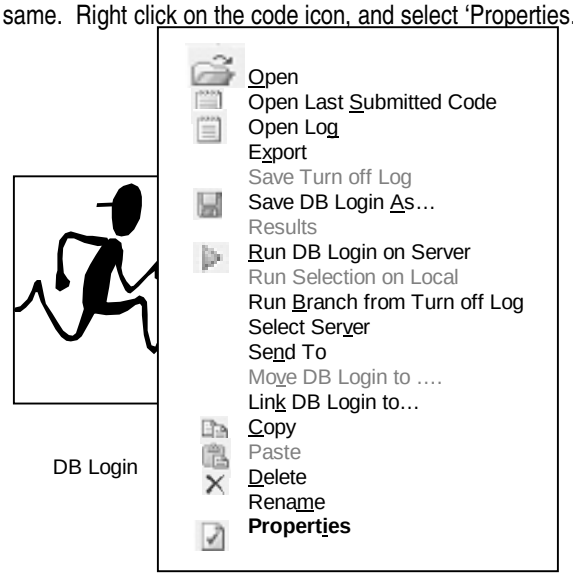



Figure 5: DB Login Code Properties

Choose "Parameters" from the Properties Screen to see those available (Figure 6):

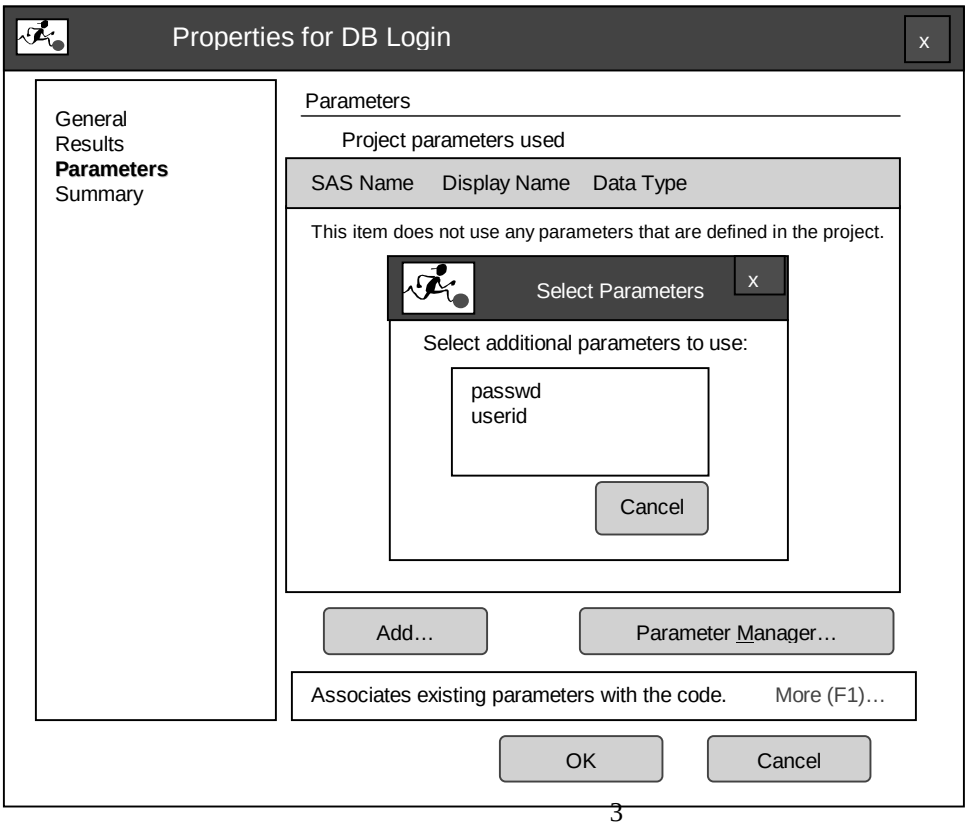



Figure 6: Parameter List

Once the parameters are added, the Properties Parameters screen should look like this (Figure 7):

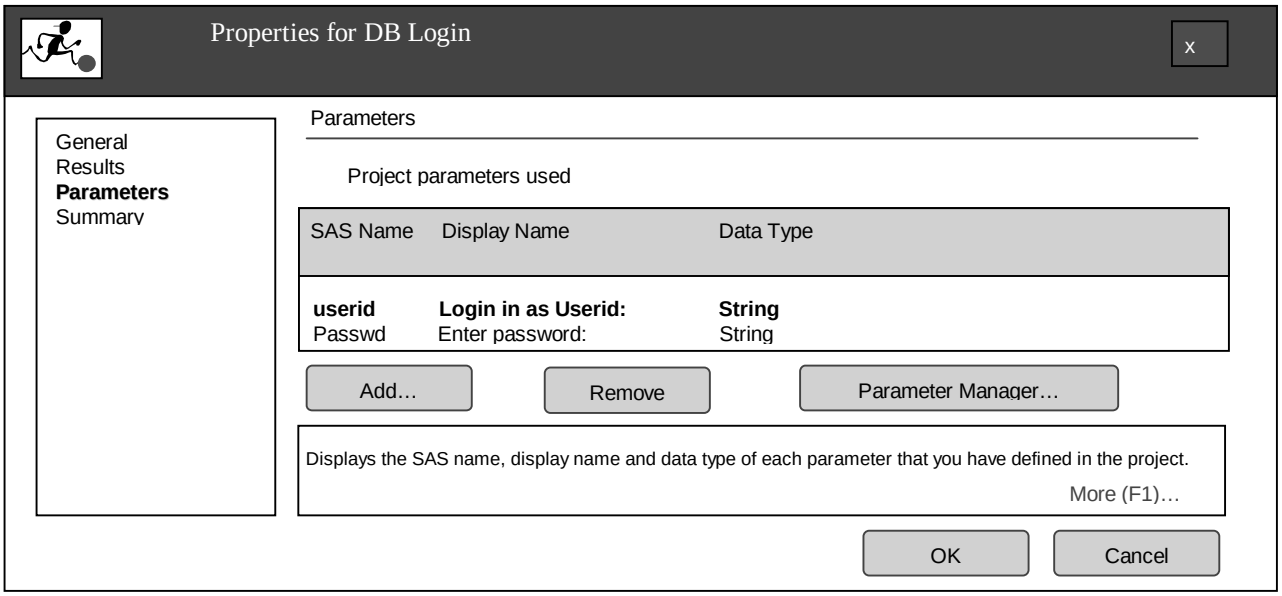



Figure 7: DB Login Parameters Screen After

Note that these parameters can be reused, added to any code needing user id and password input.

## RUNNING CODE WITH YOUR PARAMETERS

Now that you'd added the parameters to your code, you will prompted for them whenever you run it (Figure 8):

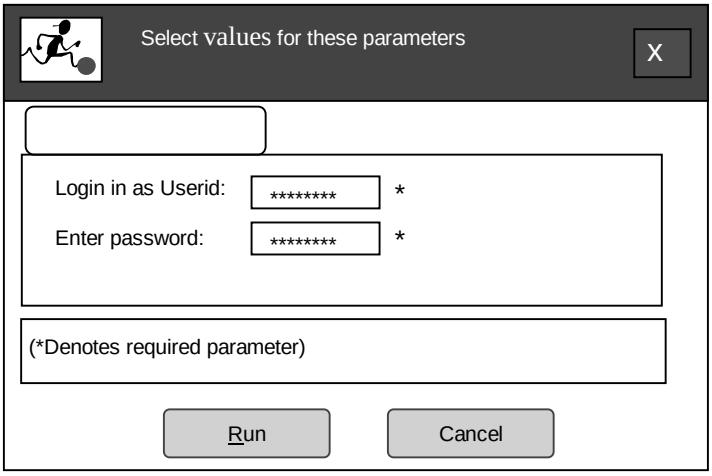



Figure 8: DB Login Prompt

Click 'Run' to execute. This should be perfectly secure – asterisks hid your user id and password, didn't they? But if you look at the log, you will see:

```
;*';*";*/;quit;run;
2  OPTIONS PAGENO=MIN;
3  %LET _CLIENTTASKLABEL=%NRBQUOTE(Database Login Example);
4  %LET _EGTASKLABEL=%NRBQUOTE(Database Login Example);
5  %LET _CLIENTPROJECTNAME=%NRBQUOTE();
6  %LET _SASPROGRAMFILE=;
7  %LET userid = %NRSTR(myid);
8  %LET passwd = %NRSTR(mypassword);
```

Not a good idea to have your user id and password showing. What if we turned the log off in the program by using option nosource? The user id and password still show because they are being set before the code ever executes:

```
;*';*";*/;quit;run;
2  OPTIONS PAGENO=MIN;
3  %LET _CLIENTTASKLABEL=%NRBQUOTE(Database Login Example);
4  %LET _EGTASKLABEL=%NRBQUOTE(Database Login Example);
5  %LET _CLIENTPROJECTNAME=%NRBQUOTE();
6  %LET _SASPROGRAMFILE=;
7  %LET userid = %NRSTR(myid);
8  %LET passwd = %NRSTR(mypassword);
9
10 ODS _ALL_ CLOSE;
11 OPTIONS DEV=ACTIVEX;
12 FILENAME EGHTML TEMP;
NOTE: Writing HTML(EGHTML) Body file: EGHTML
13  ODS HTML(ID=EGHTML) FILE=EGHTML ENCODING='utf-8' STYLE=EGDefault
13       !
13       ! NOGTITLE NOGFOOTNOTE GPATH=&sasworklocation;
14
15         %gaccessible;
16  options nosource;
```

What does work is executing a branch from a little program named "Turn off Log" consisting of just one line of code:

```
Options nosource nosymbolgen;
```

You can turn the source option back on in the DB Login program if desired:

```
options source;
Proc sql;
Connect to database connection(database=proddb tdpid=prodpid
user=&userid password=&passwd.);
Create table extract1 as
Select * from connection to database connection (select statement here).
```

Link this little program to your login code by right-clicking on the icon as you did before (Figure 9) but now choose "Link Turn Off Log to…" and select "DB Login" from the list (Figure 9):

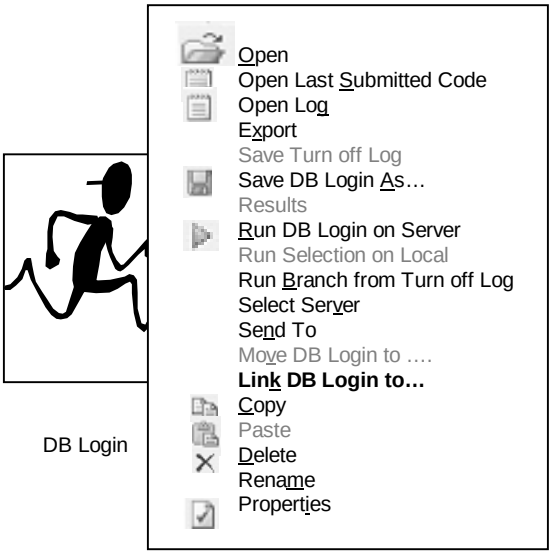
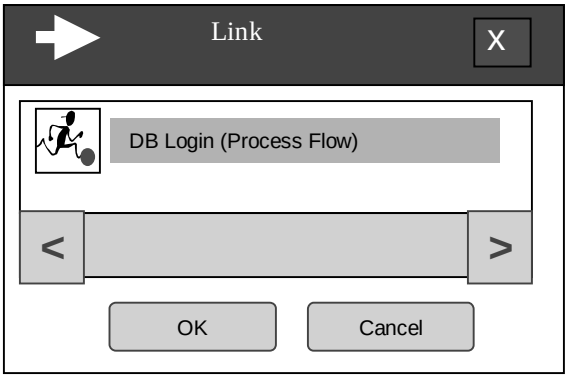



Figure 9: Link Turn Off Log to DB Login




When you run the branch from "Turn Off Log", the log shows:

```
Proc sql;
18  Connect to database connection (database=proddb tdpid=prodpid
user=&userid password=&passwd.);
19  Create table extract1 as
20  Select * from connection to database connection (select statement
here).
21
22
23  %LET _CLIENTTASKLABEL=;
24  %LET _EGTASKLABEL=;
25  %LET _CLIENTPROJECTNAME=;
26  %LET _SASPROGRAMFILE=;
20  Select * from connection to database(
```

Your user id and password are not shown. One caveat: Do not turn the symbolgen option on before connecting to the database. Otherwise your user id and password will appear in the log anyway:

```
17  Proc sql;
18  Connect to database connection (database=proddb tdpid=prodpid
18  ! user=&userid password=&passwd.);
SYMBOLGEN:  Macro variable USERID resolves to anyid
SYMBOLGEN:  Some characters in the above value which were subject to macro
quoting have been
            unquoted for printing.
SYMBOLGEN:  Macro variable PASSWD resolves to anyword
```

## MORE ON PARAMETERS

### Setting Allowable Values:

Suppose you want to run an analysis for a particular state. You can create a string variable called StateCode for use in querying a SAS® table called MailOrder. The default value will be 'IL' as marked in Figure 10. Any other values will come from a list. . One way to create this list is to simply type them in the Value List (Figure 10):

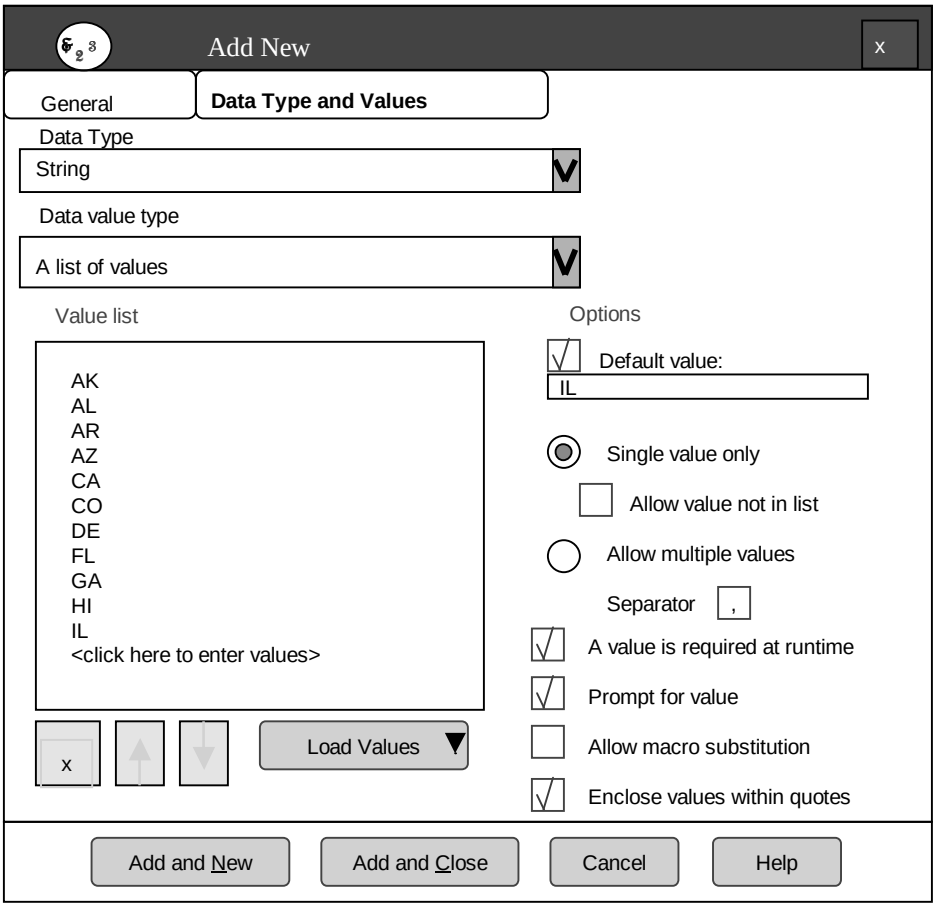



Figure 10: Value List

Typing in 50 values or more is rather tedious. Someone has a SAS dataset with all the state codes. You could load it into the list by clicking the 'Load Values' button on the 'Data Type and Values' Screen. You would be prompted for the location of the SAS® dataset containing the desired values. Once selected, you'd choose StateCode from the available columns list (Figure 11). Click "Load Values" and they are all there.

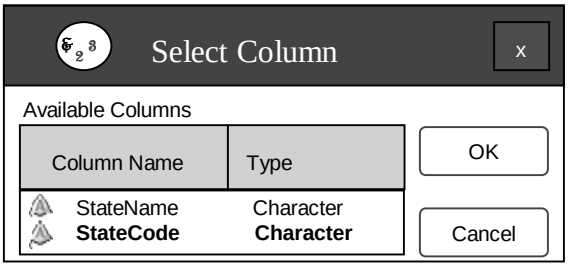



Figure 11: Load StateCode Values

Add the StateCode parameters to your code and you should be prompted every time with a drop down box like that in Figure 12:

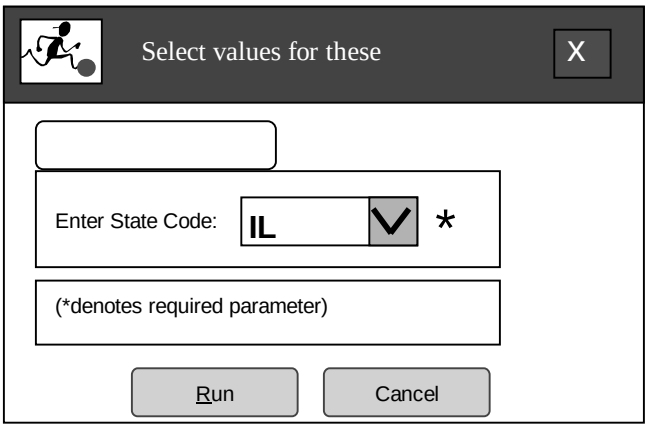



Figure 12: StateCode Drop Box

**Passing Variables to a Database Connection:**

A string variable in double quotes works just fine with SAS® code and datasets:

```
16 PROC SQL;
17 SELECT * FROM SASUSER.MAILORDER
18 WHERE ShipStateOrProvince = &STATECODE
19  ;
20  %put &sqlobs records for &statecode;
1 records for "IL"
21 run;
```

But what if you are trying to pass this parameter to a database like DB2 or Teradata, which requires string variables in single quotes? The first thing you must do is deselect the 'Enclose values within quotes' option. But then what? One option is putting the values in your list as 'AK', 'AR', 'AZ' etc. This will work whether used in a passthrough query or SAS® code.

This may prove too limiting, however. If you want to use this parameter more than once, you will find several places where quotation marks are not appropriate. You want quotes in certain situations only so you must transform the unquoted StateCode variable in your code. Maybe if you just surround it with quotes?

```
%let statecde = '&statecode';
```

Unfortunately this doesn't behave as expected. Statecde is set to '&statecode' literally.

Something that works for passthrough database queries but oddly enough not for regular SAS® Enterprise Guide® code is to use the %bquote function as such:

```
%let statecde = %bquote('&statecode');
```

This does give the variable in quotes. It passes to a database connection just fine but just attempt passing it to regular SAS® code in Enterprise Guide.

You get this error:

```
%let statecde = %bquote('&statecode');
17
18         PROC SQL;
19         SELECT * FROM SASUSER.MAILORDER
20         WHERE ShipStateOrProvince = &STATECDE
SYMBOLGEN:  Macro variable STATECDE resolves to 'IL'
21         ;
NOTE: Line generated by the macro variable "STATECDE".
21          'IL'
             _
             22
            __
            202
ERROR 22-322: Syntax error, expecting one of the following: a name, a
quoted string, a numeric constant, a datetime constant, a missing value,
(, *, +, -, ALL, ANY, BTRIM, CALCULATED, CASE, INPUT, LOWER, PUT,
SELECT, SOME, SUBSTRING, TRANSLATE, UPPER, USER.
ERROR 202-322: The option or parameter is not recognized and will be
ignored.
```

One method of setting your variable is equally effective in passthrough queries and regular SAS® code in Enterprise Guide®. This involves creating the variable in a data step, concatenating single quotes and using "call symput" to output the variable. Note all the double quotes in the assignment statement for **&statecde**:

```
data _null_;
statecde = "'"||"&statecode"||"'";
put statecode;
call symput('statecde',statecode);
run;
%put &statecode is now &statecde;
```

The log looks like:

```
16  data _null_;
17  statecde = "'"||"&statecode"||"'";
18  put statecode;
19  call symput('statecde',statecode);
20  run;
'IL'
21  %put &statecode is now &statecde;
IL is now 'IL'
22  PROC SQL;
23  SELECT * FROM SASUSER.MAILORDER
24  WHERE ShipStateOrProvince = &STATECDE
25  ;
26  %put &sqlobs records for &statecde;
1 records for 'IL'
```

**String versus Variable Name:**

You want to create a SAS® dataset from the results of this query with the naming convention state_*StateCode*. Obviously, a StateCode with single quotes will give you an error so remove them and you should be OK:

```
PROC SQL;
create table state_&statecode
as
SELECT * FROM SASUSER.MAILORDER
WHERE ShipStateOrProvince = &STATECDE
;
%put &sqlobs records for &statecde;
run;
```

But when you run it, you get this. Look a little like our %bquote results, doesn't it?

```
PROC SQL;
23          create table state_&statecode
24          as
             __
             202
NOTE: Line generated by the macro variable "STATECODE".
24           state_IL
             __
             78
ERROR 202-322: The option or parameter is not recognized and will be
ignored.
ERROR 78-322: Expecting a '.'.
25          SELECT * FROM SASUSER.MAILORDER
2                                        The SAS System         18:18

26          WHERE ShipStateOrProvince = &STATECDE;
```

But even putting in a '.' at the end of statecode gives an error:

```
24           STATE_IL
             __
             78
ERROR 202-322: The option or parameter is not recognized and will be
ignored.

ERROR 78-322: Expecting a '.'.
```

What to do? Change the variable type to 'Variable Name' in Parameter Manager. This simple change works in both creating the new SAS® dataset and as the criterion for the query:

```
24          PROC SQL;
25          CREATE TABLE STATE_&STATECODE.
SYMBOLGEN:  Macro variable STATECODE resolves to IL
26          AS
27          SELECT * FROM SASUSER.MAILORDER
28          WHERE ShipStateOrProvince = &STATECDE
SYMBOLGEN:  Macro variable STATECDE resolves to 'IL'
29          ;
NOTE: Table WORK.STATE_IL created, with 1 rows and 13 columns.
```

If it looks like we can do more with the 'Variable Name' than 'String' type, why would we even want to use 'String'? A 'String' parameter has two features not available to 'Variable Name'. One is automatic quoting and the other is input masking. In applications where these features are important, use 'String'. An interesting difference between the two is that you must pick the "Allow macro substitution" option for a 'String' variable to pick up a system variable like &sysdate while that option is not necessary with 'Variable Name'.

## CONCLUSION

Creating database logins as demonstrated makes it possible to query an outside database without having to hard code user ids and passwords anywhere, a definite security risk. For other applications, creating a list of default and valid values becomes very easy using Parameters Manager. Understanding these methods as well as delivering parameters to a passthrough query should remove one major barrier to utilizing SAS® Enterprise Guide®.

## ACKNOWLEDGEMENTS

Thank you, Joe and Paul Butkovich for reviewing this paper and presentation

## CONTACT INFORMATION

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

Patricia Hettinger
IBS
2625 Butterfield Road
Oakbrook, IL 60523
Phone: 630-309-3431
Email: hettipatra@yahoo.com