The Six Ampersand Solution

John R. Gerlach, IMS America; Plymouth Meeting, PA

Abstract

Assume you want to reinstate macro variables denoting criteria for an analysis that you performed months ago. Using Dictionary tables, made available through the SQL procedure, it is possible to collect macro variables defined for that analysis and to store them in a permanent SAS® data set containing the name of each macro variable along with its respective value. This paper explains how to preserve and, at a later time, reinstate the macro variables. As an added feature, the SAS solution lists the macro variables that have been reinstated, which requires a six ampersand solution.

The Problem

Consider an application used for the health industry that performs various analyses based on *a priori* criteria. For example, an analysis might represent patients diagnosed with Essential Hypertension from 1995 through 1996 and you are interested in specific kinds of drug therapy each patient received during a six month study period from the onset of the diagnosis. Also, other criteria are used to help ensure a proper cohort for the analysis, such as compliance issues.

Typically, a front-end to the application allows the end-user to specify the criteria needed to perform the analysis, which are stored in macro variables. However, like most front-ends, the criteria changes according to the intended analysis. Consequently, the criteria for the previous analysis must be preserved in a convenient manner in case it must be replicated or, perhaps, modified slightly.

The Idea

We need to preserve the macro variables that represent the criteria used for the analysis, along with their respective values, and to store them in a SAS data set. Then, at a later time, we will reinstate those macro variables in order to replicate the analysis.

Dictionary Tables

The SAS System affords a way of obtaining information, called metadata, about the SAS job that performed the analysis. Using SQL objects called Dictionary tables, you can obtain information about data libraries, data sets, catalogs, options, external files, and macro variables. Dictionary tables are available only through the SQL procedure; however, you can use native SAS views located in the SASHELP library.

Preserving the Macro Variables

The Data step below utilizes a view that accesses a Dictionary table and obtains the names and values of existing macro variables. By using a Data step, you can create the desired data set that contains the information needed to reinstate the macro variables at a later time. The native view VMACRO contains two variables, *scope* and *name*, which are useful in selecting the macro variables of interest. The variable *scope* allows you to exclude automatic macro variables, which are extraneous; whereas, the variable *name* allows you to exclude unwanted macro variables. Also, keep in mind that the data values in the WHERE clause are case-sensitive.

```
data project.params;
   set sashelp.vmacro(where=(scope eq 'GLOBAL'
      and name not like 'SQL%' and name
      not like'%EXIST'));
   keep name value;
run;
```

The permanent data set *params* contains two variables that are used to reinstate the criteria from a previous analysis.

name the name of the macro variable of interest.

value the value of the macro variable.

Reinstating the Macro Variables

The Data _NULL_ step below processes the permanent SAS data set *params* and reinstates the macro variables using the CALL SYMPUT routine. The SYMPUT routine has two arguments, both of which are character expressions. The first argument identifies the macro variable and the second argument contains the value to be assigned. So, obviously, the *params* data set contains precisely the information needed to reinstate the macro variables.

```
data _null_;
   set project.params;
   call symput(name,value);
run;
```

The %params Macro

The %params macro, shown below, reinstates macro variables using the CALL SYMPUT routine shown above. Also, the macro contains the added feature of listing the macro variables that have been reinstated, which requires the so-called six ampersand solution. Of

course, it would have been easier to print the contents of the *params* data set or, simply, to use the PUT statement in the Data step. However, this paper offers an exercise in the Macro Language, namely, indirect referencing of macro variables.

```
%macro params;
   data _null_;
      set project.params end=eof;
      call symput(name,value);
      call symput('var' ||
         trim(left(put(_n_,8.))),name);
      if eof then call symput
         ('nmvars',trim(left(put(_n_,8.))));
   run;
   %put Parameters:;
   %do i = 1 %to &nmvars.;
      %put &&var&i..  : &&&&&&var&i..;
      %end;
   %put;
%mend params;
```

The %params macro consists of two parts: a Data _NULL_ step and some Macro Language. The Data step reinstates the macro variables of interest using the SYMPUT routine discussed already. Also, it defines other macro variables needed to list the reinstated macro variables using two other SYMPUT routines, one which creates the macro variables **var1**, .., **var*n*** that denotes the macro variables being reinstated and the other which executes conditionally, when reading the last observation, thereby creating the macro variable **nmvars** that denotes the number of macro variables to be listed. The other part of the %params macro writes the reinstated macro variables to the SAS log by using the %DO loop and the %PUT statement along with the six ampersand solution.

How Indirect Referencing Works

In order to appreciate the six ampersand solution, it is important to understand how the macro processor resolves macro variables having two or more ampersands.

Macro variables having more than one ampersand, called indirect references, require multiple scans by the macro processor. The processor scans and resolves references for the length of the variable (usually delimited by a special character or semicolon) until all references have been resolved.

Given the following %PUT statement used in the %params macro, assume that one of the macro variables is DXCODE having the value of 4019 (i.e., the ICD-9 code for Essential Hypertension).

```
%put &&var&i..  : &&&&&&var&i..;
```

For the token **&&var&i..**, the processor reads **&&** and generates **&**; then, it reads and generates the constant text **var**; then, it reads and generates the current value of **&i.**, which denotes the *i*th iteration of the %DO loop statement.

For the first iteration of the %DO loop, the macro variable **&&var&i.** resolves to whatever the macro variable **&var1**. contains, which depends on the contents of the *params* data set obtained via the Dictionary table and the CALL SYMPUT statement that created the macro variables **&var1., …, &var*n*.** in the prior Data step. Thus,

&&var&i..
&var1.
DXCODE

The six ampersand variable (actually seven) resolves in a similar manner, however, it takes several scans to completely resolve the macro variable. That is, the macro processor reads **&&** and generates **&** (three times), reads and generates the constant text **var**, reads and generates the current value of **&i.**, as explained previously. Now, when the processor reaches the end of the token (variable), it makes a new scan, proceeding to resolve the variable **&&&var1.** anew. That is, the processor reads **&&** and generates **&**; then it reads **&var1.** and generates the contents of that macro variable, which is the name of one of the reinstated macro variables. Finally, the processor resolves the macro variable itself, thus generating its respective value. Again,

&&&&&&var&i..
&&&var1.
&DXCODE.
4019

Consequently, the %PUT statement writes the name of each macro variable and its value to the SAS log.

Conclusion

Given an application that requires a collection of criteria used for processing and reporting, it is important to preserve those criteria in order to replicate or modify an analysis. Using Dictionary tables, you can store the names and values of macro variables denoting the criteria, then recreate them at a later time.

The added feature of listing the reinstated macro variables to the SAS log offers a rigorous exercise in how the Macro Processor resolves multiple ampersands, called indirect referencing of macro variables.

Author Information

John R. Gerlach
IMS America
600 West Germantown Pike
Plymouth Meeting, PA 19462-1048
610.832.5493