

## **MACRO QUOTING RESOLVED**

**John R. Gerlach**  
**PharMark Corporation**

### **Abstract**

The purpose of macro quoting is to ensure that there is no ambiguity in the SAS<sup>R</sup> Macro Language. Ironically, unless you understand macro quoting, your code is likely to be ambiguous and prone to errors.

Macro quoting is not well understood, even by experienced macro programmers. To understand macro quoting, you must recognize ambiguous situations that can occur between the macro processor (the text generator) and the macro language (the means of communication). Consider the following example.

```
%LET prntit = PROC PRINT; RUN; ;
```

The above %LET statement is ambiguous because of the semicolons. As written, the first semicolon (after the token PRINT) will terminate the %LET statement, causing extraneous tokens (RUN;;) in your code. The macro processor needs to know that certain tokens are only text, not symbols for the macro language. Thus, you must use a quoting function, as follows.

```
%LET prntit = %STR(PROC PRINT; RUN;);
```

This paper explains macro quoting, why it is necessary and when to use it. Also, this paper discusses common functions associated with macro quoting (e.g., %STR), showing examples.

### **The Process**

The SAS System reads SAS code that is stored in an input stack one word at a time. The word scanner supplies the words (or tokens) to the word queue enroute to the SAS compiler. When the scanner detects a percent sign (%) or an ampersand (&) followed by a nonblank character (i.e., token), it triggers the macro processor. The macro processor performs its activities indicated by the SAS Macro code. Then, if any text is generated, it is placed at

the top of the input stack, ahead of the text remaining in the SAS program. The word scanner continues to supply words from the top of the input stack to the word queue. The SAS compiler collects the tokens until either a Data Step or Proc Step is completed. Then, the Step is compiled and immediately executed.

### **The Problem**

The macro processor generates text which can include tokens found in the macro language. Thus, the question arises: Should the macro processor perceive a token as text (e.g., a semicolon) or as a symbol in the macro language? Unless otherwise indicated, the macro processor interprets all tokens as part of the macro language, not as text.

### **The Answer**

To avoid ambiguous situations, the macro language implements the notion of macro quoting, which allows the user to specify whether tokens are perceived as text or as part of the macro language.

### **Ambiguous Situations**

In general, you need to quote a token only when the macro processor should not process the token as part of the macro language. An item (token) needs to be quoted when:

- the item should not be treated as an operator in the argument of an explicit or implicit %EVAL function
- a leading, trailing, or isolated blank is treated as a character, rather than ignored
- a semicolon ends a macro program statement prematurely
- a comma should not indicate a new function argument, parameter, or parameter value

- there is an unmatched single quote, double quote, or parenthesis
- there is a `%macro_name` or `&macro_name`, depending on the resolved value of the expression.

## Macro Processing

When the macro processor is activated, it interacts with the input stack and the word queue. The processor either compiles macro code or executes a macro. During compilation, the macro processor:

- compiles and stores macro statements for that particular macro
- stores all noncompiled items as text.

During the execution of a macro, the processor:

- reads text from the input stack via the word queue
- performs internal activity indicated by the macro code
- generates text.

Remember, a SAS macro is stored text that is compiled and processed by its own language, using its own word queue. Once the word scanner triggers the macro processor, the processor has control until the action is completed. The primary purpose of a macro is to generate SAS code (text).

## Quoting Functions

The macro language contains six basic functions that are used to quote text. Each quoting function tells the macro processor what to do and when to do it. Because the macro processor resolves expressions during macro execution, as well as during compilation, quoting functions are grouped according to *when* the function is processed, that is, compile or execution time. Also, these functions are grouped in terms of *what* is done with respect to the ampersand and the percent sign. Consider the following chart listing the most common quoting functions.

|             |                   |                       |
|-------------|-------------------|-----------------------|
|             | % & not<br>quoted | % &<br>quoted         |
| Compilation | %STR              | %NRSTR                |
| Execution   | %QUOTE<br>%BQUOTE | %NRQUOTE<br>%NRBQUOTE |

If an item affects the way the macro processor constructs macro program statements, you must quote the item during macro compilation using either %STR or %NRSTR functions as needed. If the macro processor resolves expressions during execution, causing problems with respect to the macro language, you must use the other quoting functions, such as %QUOTE. Using an execution-time quoting function in a macro definition ensures that the macro processor accepts resolved values that cause problems. The NR prefix denotes No-Rescan with respect to the ampersand and the percent sign.

The following text briefly explains each quoting function used in the SAS Macro Language.

## The %STR Function

Purpose: quotes constant text.

Situations:

- to treat a semicolon as text rather than as part of a macro statement
- to make blanks significant that otherwise would be ignored
- to read or write an unmatched quote or parenthesis.

Examples:

```
%MACRO stooges(default=
    %STR(Moe, Larry, Curley));

%LET pointr = %STR( == > );

%PUT %STR(I.%) Chapter One);

%LET brat = %STR("Scarlet O%'Hara");;
```

### **The %NSTR function**

**Purpose:** quotes constant text including % and & so that the macro processor does not recognize either signs.

**Situation:**

- same as for the %STR function except that the argument contains a macro variable reference or a macro invocation.

**Examples:**

```
%LET credits = %NSTR(Moe,Larry&Curley);
```

```
%LET fnote = %NSTR(72%profit);
```

### **The %QUOTE Function**

**Purpose:** quotes resolved value during macro execution.

**Situation:**

- you need to quote the result of a resolved argument.

**Example:**

```
%MACRO sumit;  
  %IF %QUOTE(&arg.) ^=  
    %THEN SUM(OF &arg.);  
  %ELSE 0;  
%MEND sumit;  
%sumit(var1-var3)
```

Without using the %QUOTE function in the above example, the macro processor produces an error message for the invalid %IF expression, as shown.

```
%IF var1-var3 ^=
```

If it is not possible to mark an unmatched quote or a parenthesis, use %BQUOTE or %NRBQUOTE.

### **The %NRQUOTE Function**

**Purpose:** Same as %QUOTE, but quotes the & and % signs as well.

**Situations:**

- same as the %QUOTE function except that the argument contains a macro variable reference or a macro invocation.

**Example:**

```
%LET tok = %NRQUOTE(J&P Construction);  
%MACRO ex_nrq;  
  %PUT The value of tok is: &tok.;  
%MEND ex_nrq;  
%ex_nrq;
```

Without using the %NRQUOTE function in the above example, the macro processor would attempt to resolve &P. Using this function produces warning messages in the SASLOG.

### **The %BQUOTE Function**

**Purpose:** quotes a resolved value, including unanticipated special characters (specifically parentheses and quote marks), but not % and &.

**Situation:**

- the resolved value contains unmatched quotes or parentheses.

**Example:**

```
%MACRO ex_bq;  
  %GLOBAL place;  
  %PUT Enter location of meeting;  
  /* Enter: John's office */  
  %INPUT;  
  %IF %BQUOTE(&SYSBUFR.) NE ;  
    %THEN %PUT Location Entered.;  
    %ELSE %PUT Specify location!;  
%MEND ex_bq;  
%ex_bq
```

## The %NRBQUOTE Function

**Purpose:** quotes a resolved value, unanticipated special characters, including % and & signs.

**Situation:**

- the resolved value contains a macro reference or a macro invocation that may generate an unmatched quote or parenthesis.

**Example:**

```
%MACRO ex_nbq;
  %GLOBAL place;
  %PUT Enter location of meeting;
  /* Enter: Use R&D Lab in John's office */
  %INPUT;
  %IF %BQUOTE(&SYSBUFFR.) NE ;
    %THEN %PUT Location Entered;
    %ELSE %PUT Specify location!;
%MEND ex_nbq;
%ex_bq
```

In Version 6, the %NRBQUOTE function quotes mnemonic operators such as LE, AND, GT, in addition to special characters. A warning message is written to the SASLOG, but the macro processor does not attempt to resolve the reference.

## Other Quoting Functions

The SAS System unquotes quoted items at the so-called macro boundary inside the word queue for final processing by the SAS compiler. However, there are instances when you need to unquote the tokens prior to reaching the macro boundary. The %UNQUOTE function serves this purpose.

The %UNQUOTE function is useful when you need to use a quoted token with its original meaning at a later time inside the macro.

The %SUPERQ function causes the macro processor to treat the value of a macro variable as text, preventing resolution of the macro variable's value. The %SUPERQ function uses only the macro variable name, not the reference to the macro variable whose value you are quoting. For example,

assume that you want to examine a value assigned by the CALL SYMPUT routine such that the value may contain an ampersand or a percent sign, as follows.

```
DATA _NULL_;
  CALL SYMPUT('mvar1','Clinton&Gore');
  CALL SYMPUT('mvar2','%macro m;');
RUN;

%LET chkmv1 = %SUPERQ(mvar1);
%LET chkmv2 = %SUPERQ(mvar2);
```

```
%PUT The value of &chkmv1. and &chkmv2.;
```

## Conclusion

Quoting functions are necessary for any rigorous application involving the SAS Macro Language. Also, quoting functions may be the solution to a problem with existing macro code. The user must learn to recognize ambiguous situations and to apply the appropriate quoting function in order to write better macro code and to solve nasty macro problems.

SAS<sup>®</sup> is a registered trademark of SAS Institute.