Paper CT06

RESOLVE – A Lesser-known Powerful SAS Data Step Function for Retrieving Macro Variable Values

Navneet Agnihotri, Syneos Health®, Gurgaon, India
Sumit Pratap Pradhan, Syneos Health, Gurgaon, India

ABSTRACT

The SAS® macro facility is an extremely powerful tool of the SAS System. The SAS user can create macro variables from and within a data step by CALL SYMPUT routine and the values of the macro variable created in a data step can be retrieved by SAS data step functions SYMGET and RESOLVE.

One would have very often encountered SYMGET function employed in macro program but very rarely could find reference of RESOLVE function in any macro program. Though RESOLVE function performs similar tasks as SYMGET function, it is however much powerful than SYMGET. It accepts a wider variety of arguments than SYMGET and resolves any macro expression unlike SYMGET which resolves only a single macro variable. This paper will compare and demonstrate the usefulness and relative ease of the lesser-known RESOLVE function compared to SYMGET function in retrieving values of macro variable(s).

INTRODUCTION

The data step interface consists of 2 routines and 2 functions that can create, resolve, and execute macros or macro variables:

1) CALL SYMPUT Routine – Allows the creation of SAS macro variables from within a SAS data step using either data step variable values or data set information.
2) CALL EXECUTE Routine – Allows the execution of a SAS macro from within a SAS data step or any SAS code, which also includes the possibility for a macro call.
3) SYMGET Function – Allows retrieving of macro variables into data step variable at program run-time.
4) RESOLVE Function – Allows retrieving of macro variables references and data step variables containing macro references at compilation time or execution time.

RESOLVE

The RESOLVE function is a data step function that returns the resolved value of the argument after the argument has been processed by the macro facility. It is used to resolve macro variable references, macros (passed as argument to the function) and data step variables containing macro references. It can also control whether that resolution occurs at compilation time or execution time, making it far more powerful than SYMGET. The type of quotation marks used determines when a macro variable is resolved. If double quotes are used, then resolution takes place at compilation time and the value returned can only be the value held at the outset. If single quotes are used, then resolution is delayed until execution time (i.e. when the data step is processing the data) and the current value for the macro variable is returned.

If the RESOLVE function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200 bytes.

SYNTAX

The syntax for using a RESOLVE function is as below –

RESOLVE (argument)

where argument can be one of the following items:

- a text expression. When a macro variable value contains a macro variable reference, RESOLVE attempts to resolve the reference. If argument references a nonexistent macro variable, RESOLVE returns the unresolved reference.

```
demo = resolve ('&city') ;
```

Referencing a macro variable

```
demo = resolve ('%city') ;
```

Referencing a macro invocation

- the name of a data step variable whose value is a text expression.

```
demo1 = '&locate' ;
demo2 = resolve (demo1) ;
```

- a character expression that produces a text expression for resolution by the macro facility.

```
demo = resolve('&city') || strip(code) ;
```

If RESOLVE cannot locate the macro variable or macro identified by the argument, it returns the argument without resolution and the macro processor issues a warning message. You can create a macro variable with the SYMPUT routine and use RESOLVE to resolve it in the same data step.

EXAMPLE 1

The below example shows RESOLVE used with a macro variable reference, a macro invocation, and a data step variable whose value is a macro invocation. The usage of single quotation mark signifies that the values of respective argument are being resolved at run-time.

```
%let event = PHUSE EU Connect ;

%macro evntdy ;

    Tuesday

%mend evntdy ;

data demo ;
       when = '%evntdy' ;
       demo1 = resolve ('&event') ;      /*** macro variable reference ***/
       demo2 = resolve ('%evntdy') ;     /*** macro invocation ***/
       demo3 = resolve (when) ;          /*** data step variable with macro invocation ***/

       put demo1 = ;
       put demo2 = ;
       put demo3 = ;

run ;
```

When the above program is submitted in SAS, the following message is written to the SAS log.

```
demo1=PHUSE EU Connect
demo2=Tuesday
demo3=Tuesday
```

In the above program, variable DEMO1 references a macro variable EVENT which carries string value **PHUSE EU Connect**. The RESOLVE function resolves the values of macro variable EVENT and assigns it to variable DEMO1. Variable DEMO2 takes macro invocation EVNTDY as argument to RESOLVE function and resolves to Tuesday. Variable DEMO3 references a data step variable WHEN which contains value as “%evntdy”. This is a reference to macro invocation EVNTDY and similar to variable DEMO2 processing. Hence, in a similar manner, variable DEMO3 is assigned a value **Tuesday** by the RESOLVE function.

EXAMPLE 2

In the below example, let's see the magic of RESOLVE function when it resolves arguments at compilation and execution time respectively. %let is used to set a macro variable value at compilation time and call SYMPUT is used to change that value during execution time.

```
%let eventday = Tuesday ;

data demo ;

        call symput ('eventday', 'Monday') ;

        demo1  = resolve ("&eventday") ;
        demo2 = "&eventday" ;

        put "Compilation Time :" ;
        put demo1 = ;
        put demo2 = ;


/*************************************************************************/

        demo3 = resolve ('&eventday') ;
        demo4 = symget ('eventday') ;

        put "Execution Time :" ;
        put demo3 = ;
        put demo4 = ;

run ;
```

Execution Time

Double quotations will resolve the values at compilation time

Single quotations will resolve the values at execution time

When the above program is submitted in SAS, the following message is written to the SAS log.

```
Compilation Time :
demo1=Tuesday
demo2=Tuesday
```

```
Execution Time :
demo3=Monday
demo4=Monday
```

From the above output, the variables DEMO1 and DEMO2 were assigned values during compilation time and hence they carry the value as **Tuesday**. However, variables DEMO3 and DEMO4 were assigned values during execution time and hence value **Monday** has been assigned to DEMO3 and DEMO4 which was created by CALL SYMPUT routine and assigned to macro variable EVENTDAY.

EXAMPLE 3

Uniquely to the RESOLVE function, macro references within the text of data step variables can also be resolved. The below example illustrates this feature of RESOLVE function.

```
%let evntday = Tuesday ;

%macro eventday ;

      Day inside the macro is &evntday

%mend eventday ;

data demo ;

      call symput ('evntday', 'Monday') ;

      demo1 = resolve ("&evntday") ;
      demo2 = resolve("%eventday") ;
      text1 = "Data variable Day is &evntday." ;
      demo3 = resolve(text1) ;
      text2 = "Data variable %eventday" ;
      demo4 = resolve(text2) ;

      put "Compilation Time :" ;
      put demo1 = ;
      put demo2 = ;
      put demo3 = ;
      put demo4 = ;

      /*************************************************************************/

      demo5 = resolve('&evntday') ;
      demo6 = resolve('%eventday') ;
      text3 = 'Data variable Day is &evntday.' ;
      demo7 = resolve(text3) ;
      text4 = 'Data variable %eventday.' ;
      demo8 = resolve(text4) ;

      put "Execution Time :" ;
      put demo5 = ;
      put demo6 = ;
      put demo7 = ;
      put demo8 = ;

run ;
```

Execution Time

Compilation Time

Execution Time

When the above program is submitted in SAS, the following message is written to the SAS log.

```
Compilation Time :
demo1=Tuesday
demo2=Day inside the macro is Tuesday
demo3=Data variable Day is Tuesday
demo4=Data variable Day inside the macro is Tuesday
```

```
Execution Time :
demo5=Monday
demo6=Day inside the macro is Monday
demo7=Data variable Day is Monday
demo8=Data variable Day inside the macro is Monday
```

In the above program, the difference to note here is the reference to macro variable EVNTDAY and macro invocation EVENTDAY within the text of data step variables TEXT1, TEXT2, TEXT3 and TEXT4. The RESOLVE function while retrieving the value of the data step variable TEXT1 and TEXT3 and assigning it to DEMO3 and DEMO7 variables, it first resolves the value of the macro variable EVNTDAY during the compilation and execution time respectively and then assigns the complete resolved value **Data variable Day is Tuesday** to the variable DEMO3 during compilation time and value **Data variable Day is Monday** to the variable DEMO7 during execution time. In a similar manner, for the variables TEXT3 and TEXT4, it invokes the macro EVENTDAY, resolves the value and assigns the resolved value **Data variable Day inside the macro is Tuesday** to the variable DEMO4 during compilation time and value **Data variable Day inside the macro is Monday** to the variable DEMO8 during execution time.

RESOLVE COMPARED TO SYMGET

- RESOLVE retrieves the value both at compilation and execution time while SYMGET retrieves only at execution time.

- RESOLVE accepts a wider variety of arguments than the SYMGET function accepts. SYMGET resolves only a single macro variable, but RESOLVE resolves any macro expression. Using RESOLVE might result in the execution of macros and resolution of more than one macro variable.

- When a macro variable value contains an additional macro variable reference, RESOLVE attempts to resolve the reference, but SYMGET does not.

- In case of unresolved macro variables, SYMGET will return a missing value whereas RESOLVE will return the unresolved macro reference.

CONCLUSION

As illustrated above, the RESOLVE function is a powerful function that could fulfil your requirements and resolve your issues more easily. The times when RESOLVE is essential may be relatively rare, but it is crucial for simple solutions at these times. The illustrations given above could not have been solved in one step without writing a relatively complex parsing routine.

ACKNOWLEDGMENT

The authors would like to acknowledge the “Material Review Board” group of Syneos Health and “Coding Tips and Tricks Stream Chairpersons of PHUSE” for their thoughtful review of this manuscript.

CONTACT INFORMATION

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

Navneet Agnihotri
Syneos Health, Principal Statistical Programmer
DLF Downtown, DLF City Phase 3 Road, Sector 25A, Gurgaon - 122002, Haryana, India
E-mail: navneet.agnihotri@syneoshealth.com
LinkedIn: https://www.linkedin.com/in/navneet-agnihotri-78540553/

Sumit Pratap Pradhan
Syneos Health, Manager, Statistical Programming
DLF Downtown, DLF City Phase 3 Road, Sector 25A, Gurgaon - 122002, Haryana, India
E-mail: sumit.pradhan@syneoshealth.com
LinkedIn: https://www.linkedin.com/in/sumit-pradhan-71133345/