



PHUSE EU Connect 2022 (Presentation – CT06)

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

Navneet Agnihotri, Principal Statistical Programmer, Syneos Health®

Sumit Pratap Pradhan, Manager (Statistical Programming), Syneos Health

Introduction

- The SAS® macro facility is an extremely powerful tool of the SAS System.
- One would have very rarely encounter reference of RESOLVE function in any macro program.
- RESOLVE function performs similar tasks as SYMGET function, it however accepts a wider variety of arguments than SYMGET and resolves any macro expression unlike SYMGET which resolves only a single macro variable.

Features of RESOLVE Function

- 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. If single quotes are used, then resolution is delayed until execution time).
- 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.

General Syntax of RESOLVE Function

The syntax for using a RESOLVE function is as below –

RESOLVE (argument)

where argument can be one of the following items:

- ❖ a text expression.

```
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) ;
```

Example

```
%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

Output

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

RESOLVE compared to SYMGET

- RESOLVE retrieves the value both at compilation and execution time while SYMGET retrieves values only at execution time.
- RESOLVE retrieves values from any macro/text expression (including macro invocation) while SYMGET resolves only a single 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

Contact Information

Navneet Agnihotri

Principal Statistical Programmer

navneet.agnihotri@syneoshealth.com

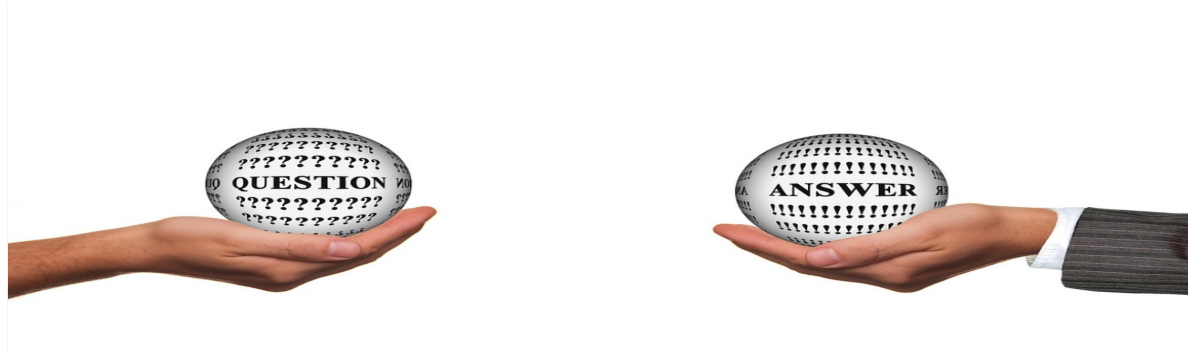
<https://www.linkedin.com/in/navneet-agnihotri-78540553/>

Sumit Pratap Pradhan

Manager, Statistical Programming

sumit.pradhan@syneoshealth.com

<https://www.linkedin.com/in/sumit-pradhan-71133345/>



Shortening the Distance from Lab to Life[®].

