



The Ampersand (&) Challenge Single, Double or more?

Amar Nayak
Sequensys Ltd
Freelance SAS® Consultant
Maidenhead, UK



Aim and Purpose of the Paper

While developing or reviewing macro code, most of us would have gone through similar emotions looking at the use of one or more ampersand (“&”) in relation to macro variables

& ➡ && ➡ &&&&.....



The aim and purpose of this presentation (and the paper) is

- To explain the macro variable resolution process via graphical representation(s) that any SAS® Programmer shall easily understand
- Encourage SAS® programmers to such approach in building macro variables in their programs



Topics covered in the Presentation

- Basics rules of macro resolution process
- Scenario 1 - “&” Pyramid
- Scenario 2 - Example 1 Direct Macro Variable Reference
- Scenario 2 - Example 2 Indirect Macro Variable Reference
- Scenario 2 - Example 3 Multi Macro Variable Combination



Understanding the Basic Rules

The following rules of the macro variable resolution process should be clearly understood

- 1) The macro processor reads a macro variable from left to right (i.e. forward scanning).
- 2) For multiple ampersands, the re-scan rule takes effect until no more ampersands are left to resolve i.e the macro variable resolves its final value OR error messages have been issued
- 3) 'Two' ampersands result into 'One' ampersand (&& → &) during the macro resolution process

These rules are explained in greater detail with different types of scenarios in dealing with single, double or multiple (>2) ampersand (“&”) combination



Scenario 1 – “&” Pyramid

To begin with, let me illustrate an example of defining four macro variables separately and how multiple ampersands deal with these uniquely defined macro variables `** (n) **`.

```
options symbolgen;
```

```
%LET BOOK = STORY;    ** (1) **;  
%LET STORY = FUNNY;    ** (2) **;  
%LET FUNNY = TWIST;    ** (3) **;  
%LET TWIST = CLIMAX;   ** (4) **;
```

```
%put &BOOK;  
%put &&&BOOK;  
%put &&&&&&BOOK;  
%put &&&&&&&&&&&&&&&&&&BOOK;
```

$$y = 2^n - 1$$



Y: Number of ampersands pre-fixed to the macro variable

M: Macro variable to be resolved

R: Final resolved value

Y	M		R
1	&BOOK	→	STORY
2	&&BOOK	→	STORY
3	&&&BOOK	→	FUNNY
4	&&&&BOOK	→	STORY
5	&&&&&BOOK	→	FUNNY
6	&&&&&&BOOK	→	FUNNY
7	&&&&&&&BOOK	→	TWIST
8	&&&&&&&&BOOK	→	STORY
9	&&&&&&&&&BOOK	→	FUNNY
10	&&&&&&&&&&BOOK	→	FUNNY
11	&&&&&&&&&&&BOOK	→	TWIST
12	&&&&&&&&&&&&BOOK	→	FUNNY
13	&&&&&&&&&&&&&BOOK	→	TWIST
14	&&&&&&&&&&&&&&BOOK	→	TWIST
15	&&&&&&&&&&&&&&&BOOK	→	CLIMAX

In mathematical terms,

- 'n' is the number of macro variables that are defined
- 'y' is the number of ampersands that are required until the macro variable successfully resolves to its final value



(4) &&&&&&&&&&&BOOK¹ → &&&&&STORY² → &&&FUNNY³ → &TWIST⁴ → CLIMAX

In this example after executing the macro variable, the SAS® log will appear as follows

After 1st iteration

&& && && && && && && &BOOK

1 2

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

& & & & & & STORY

① During the first read of the macro variable from left to right, seven pairs of double ampersands result into one (&& → &) and the rest &BOOK resolves to its defined value as STORY



The “Inside Story” (contd....)

SYMBOLGEN: && resolves to &.
SYMBOLGEN: && resolves to &. ③
SYMBOLGEN: && resolves to &.
SYMBOLGEN: Macro variable STORY resolves to FUNNY

After 2nd iteration

① ②
&&&&&&&STORY
↓ ↓ ↓ ↓
& & & FUNNY

② During the second read, three pairs of double ampersands result into one (&& → &) and the rest &STORY resolves to its defined value as FUNNY

SYMBOLGEN: && resolves to &. ③
SYMBOLGEN: Macro variable FUNNY resolves to TWIST

After 3rd iteration

① ②
&&&FUNNY
↓ ↓
& TWIST

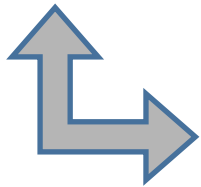
③ During the third read, first two ampersands result into one (&& → &) and the rest &FUNNY resolves to its defined value as TWIST



The “Inside Story” (contd....)

SYMBOLGEN: Macro variable TWIST resolves to CLIMAX
CLIMAX

After 4th iteration to its
final value highlighted



&TWIST ① ②
↓
CLIMAX

④

During the fourth read, &TWIST resolves to its defined value as CLIMAX



Scenario 2 - Example 1

Direct Macro Variable Reference

This example lists the past conference details
using directly referenced macro variables

```
*****;  
*** DEFINE MACRO VARIABLES ***;  
*****;  
%let phuscit1 = Manchester;  
%let phuscntr1 = England;  
%let yr1 = 2008;  
  
%put PhUSE Conference was held at &phuscit1 (&phuscntr1) in &yr1;  
  
%let phuscit2 = Basel;  
%let phuscntr2 = Switzerland;  
%let yr2 = 2009;  
  
%put PhUSE Conference was held at &phuscit2 (&phuscntr2) in &yr2;
```

Here the conference details
of any year gets displayed
through a combination of
**free text and directly
referenced macro variables**

A snippet of the SAS® log after executing the SAS® program above is as follows

```
SYMBOLGEN:  Macro variable PHUSCITY1 resolves to Manchester  
SYMBOLGEN:  Macro variable PHUSCNTR1 resolves to England  
SYMBOLGEN:  Macro variable YR1 resolves to 2008  
PhUSE Conference was held at Manchester (England) in 2008
```

```
SYMBOLGEN:  Macro variable PHUSCITY2 resolves to Basel  
SYMBOLGEN:  Macro variable PHUSCNTR2 resolves to Switzerland  
SYMBOLGEN:  Macro variable YR2 resolves to 2009  
PhUSE Conference was held at Basel (Switzerland) in 2009
```



Scenario 2 - Example 2

Indirect Macro Variable Reference

This example lists the past conference details using indirectly referenced macro variables with %DO loop

```
*****;
*** CREATING DUMMY DATA ***;
*****;
data phuse_city;
  input yr city $11. country $12. @
  datalines;
    2008 Manchester England
    2009 Basel Switzerland
    2010 Berlin Germany
    2011 Brighton England
    2012 Budapest Hungary
    2013 Brussels Belgium
    2014 London England
    2015 Vienna Austria
  ;
run;
```

```
*****;
*** DEFINE MACRO VARIABLES ***;
*****;
data _null_;
  set phuse_city end=eod;

  /* CREATE MACRO VARIABLES STORING START YEAR VALUE */
  if _n_ = 1 then call symput("startyr", compress(put(yr, best.)));

  /* CREATE MACRO VARIABLES STORING CURRENT YEAR VALUE */
  if eod then call symput("curryr", compress(put(yr, best.)));

  /* CREATE FIRST SET OF SEVEN MACRO VARIABLES STORING YEAR VALUES */
  call symput("yr"||compress(put(yr, best.)), compress(put(yr, best.)));

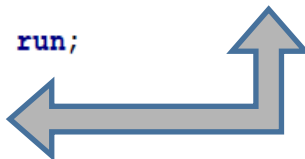
  /* CREATE SECOND SET OF SEVEN MACRO VARIABLES STORING CITY VALUES
  CORRESPONDING TO EACH YEAR */
  call symput("phuscit"||compress(put(yr, best.)), compress(city));

  /* CREATE THIRD SET OF SEVEN MACRO VARIABLES STORING COUNTRY
  VALUES CORRESPONDING TO EACH CITY */
  call symput("phuscnt"||compress(put(yr, best.)), compress(country));

run;
```

The macro variables (and series) defined are

- 1) &startyr
- 2) &curryr
- 3) &yr2008 - &yr2015
- 4) &phuscit2008 - &phuscit2015
- 5) &phuscnt2008 - &phuscnt2015





Scenario 2 - Example 2 (contd....)

```
*****;  
*** CREATE A MACRO CODE WITH %DO LOOP TO OUTPUT PAST CONFERENCE DETAILS ***;  
*** USING MACRO VARIABLES ASSOCIATED WITH DOUBLE AMPERSANDS (&&) ***;  
*****;  
%macro phuse_city ();  
  
  %do i = &startyr %to &curryr;  
  
    %put The PhUSE Conference was held at &&phuscity&i (&&phuscntr&i) in  
      &&yr&i;  
  
  %end;  
  
%mend phuse_city;  
  
*****;  
*** EXECUTE MACRO ***;  
*****;  
%phuse_city;
```

Here the conference details of any year gets displayed through a combination of **free text** and **indirectly referenced macro variables**

A snippet of the SAS® log after executing the improvised SAS® program is as follows

```
MLOGIC(PHUSE_CITY): Beginning execution.  
SYMBOLGEN: Macro variable STARTYR resolves to 2008  
SYMBOLGEN: Macro variable CURRYR resolves to 2015  
MLOGIC(PHUSE_CITY): %DO loop beginning; index variable I; start value is 2008; stop value is 2015;  
MLOGIC(PHUSE_CITY): %PUT The PhUSE Conference was held at &&phuscity&i (&&phuscntr&i) in &&yr&i  
SYMBOLGEN: && resolves to &. (1)  
SYMBOLGEN: Macro variable I resolves to 2008 (2)  
SYMBOLGEN: Macro variable PHUSCITY2008 resolves to Manchester (1)  
SYMBOLGEN: && resolves to &. (1)  
SYMBOLGEN: Macro variable I resolves to 2008 (2)  
SYMBOLGEN: Macro variable PHUSCNTR2008 resolves to England (1)  
SYMBOLGEN: && resolves to &. (1)  
SYMBOLGEN: Macro variable I resolves to 2008 (2)  
SYMBOLGEN: Macro variable YR2008 resolves to 2008  
The PhUSE Conference was held at Manchester (England) in 2008
```

The macro resolution steps indicated by (1) happen simultaneously followed by (2) resulting in the final display of conference text output



Scenario 2 - Example 2 (contd....)

The “Inside Story”

The PhUSE Conference was held at **&&phuscit&i** (**&&phuscnt&i**) in **&&yr&i**

After First Iteration



The PhUSE Conference was held at **&phuscit&2008** (**&&phuscnt&i**) in **&&yr&i**

After Second Iteration



The PhUSE Conference was held at **Manchester** (**&&phuscnt&i**) in **&&yr&i**

After Third Iteration



The PhUSE Conference was held at **Manchester** (**&phuscnt&2008**) in **&&yr&i**

After Fourth Iteration



The PhUSE Conference was held at **Manchester** (**England**) in **&&yr&i**

After Fifth Iteration



The PhUSE Conference was held at **Manchester** (**England**) in **&yr&2008**

After Sixth Iteration



The PhUSE Conference was held at **Manchester** (**England**) in **2008**



 context
PhUSE Conference was held at Manchester (England) in 2008
PhUSE Conference was held at Basel (Switzerland) in 2009
PhUSE Conference was held at Berlin (Germany) in 2010
PhUSE Conference was held at Brighton (England) in 2011
PhUSE Conference was held at Budapest (Hungary) in 2012
PhUSE Conference was held at Brussels (Belgium) in 2013
PhUSE Conference was held at London (England) in 2014
PhUSE Conference was held at Vienna (Austria) in 2015



13



After First Iteration





Conclusion

- There is no fixed approach for the use of multiple ampersands (“&”)
- The use of it depends of what the SAS® macro program is trying to achieve in relation to the programming specs
- Handling of indirectly referenced macro variables with multiple “&” requires careful planning to achieve the desired results
- Last but not the least, I hope the paper encourages SAS® programmers to take the multiple “&” challenge

