MWSUG2016-BB07

Be Prompt – Part II!

Advanced Prompting Techniques in SAS® Enterprise Guide

Ben Cochran, The Bedford Group, Raleigh, NC

ABSTRACT

With some fairly simple (and some not so simple) programming behind the scenes, you can take basic Enterprise Guide prompts, that are already powerful, and make them even more robust. This paper shows many examples on how to do this as well as how to overcome some challenges to 'out of the box' prompts. Things like 'verifying' prompt selections and how to cut and paste values into prompts will be illustrated in this presentation.

INTRODUCTION

Suppose we want to prompt for a series of values, both individual values and ranges. EG has prompts that can do either, but not both at the same time. Well, not right out of the box anyway. The first example will look at solving this problem. Example 2 looks at issues surrounding the verification of values entered into prompts. The third example examines issues surrounding the implemetation of **And/Or** prompts. And finally, the last example addresses ways that you can cut and paste values into prompts. The data used in this presentation comes from the SASHELP. CLASS dataset and is used for its simplicity.

EXAMPLE 1: ENTERING A RANGE OF VALUES

In this section, we are going to create a prompt that allows you to enter individual values as well as a range of values. So, imagine a prompt that looks like this after you have entered some values.

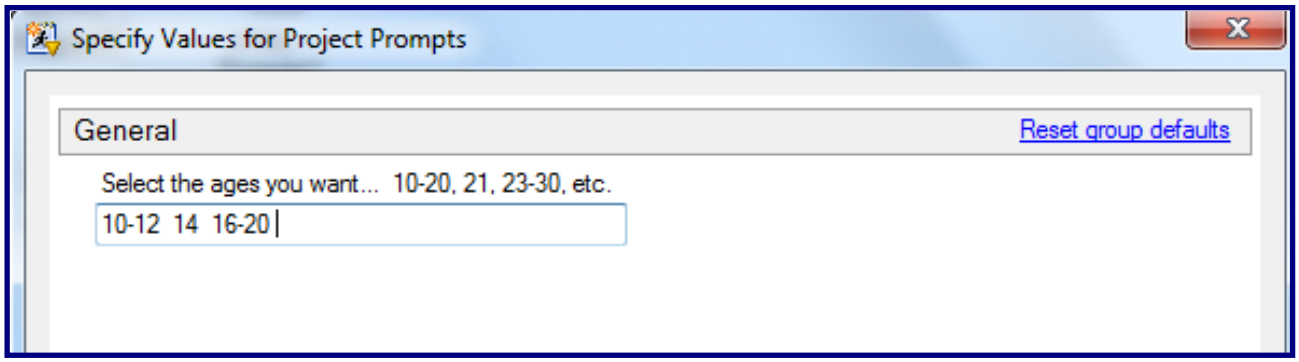


Figure 1. Prompting for a Range of Values.

Suppose we are looking for a range of ages, but we don't want to enter every single age between 10 and 20. In the prompt shown above, we we want to get all ages between 10 and 20 except 13 and 15. We will create this prompt and call it **Age_2.** This is done by taking the following steps:

1. Open the Prompt Manager in Enterprise Guide and select **Add.**
2. When the Add New Prompt window opens, supply the name **Age_2,** and specify what you want as the displayed text.
3. **Select the Prompt Type and Values** tab.
4. Use the default value of **Text** for the **Prompt type.**
5. Make the **Method for populating prompt: User enters values.**
6. Make the **Number of values: Single value.**
7. **Text type** can be **Single line.**

When finished, the tab should look like Figure 2.

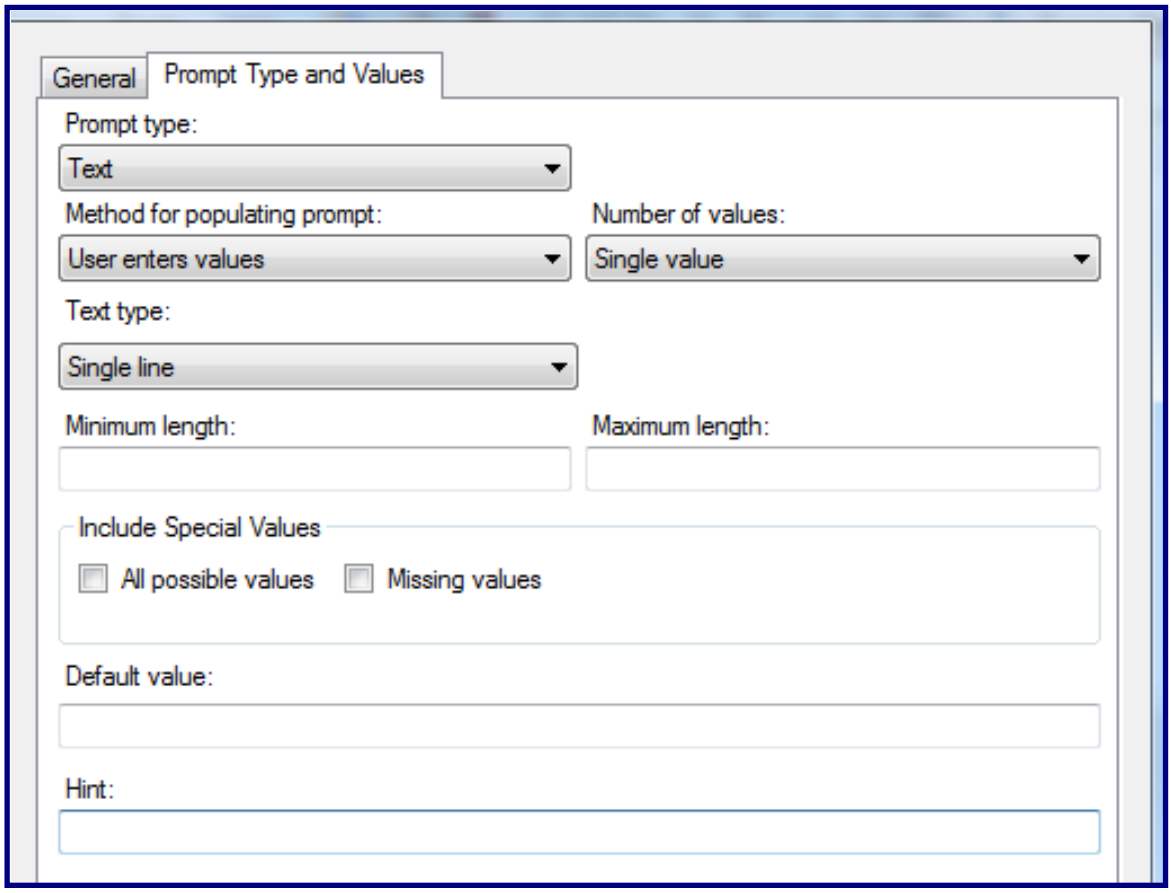


Figure 2.

Select **OK** to complete the building / editing of the prompt**.** This prompt creates a **macro variable** called **Age_2.** The next step is to write a program that takes these prompt values and translates them into values we can place in a **WHERE** statement**.** The program is broken down into several steps**:**

1. Find the number of ranges and /or individual numbers**.**
2. Convert ranges into consecutive numbers.
3. Convert all values represented by the prompt selection to a comma separated list of values.
4. Use this list of values to build a WHERE statement
5. Get the list of valid values (from a table)**.**
6. Do some data manipulation**.**
7. Compare the values chosen in the prompt to the **Valid** values**.**

```
* Step 1:  Find the number of individual values and/or ranges.     *;
*          Values and Ranges can be separated by blanks or commas. *;
*-------------------------------------------------------------------*;
   %let prompt = &age_2;
   data _null_;
           c=count("&prompt", ',');
           if c=0 then c=count("&prompt", ' ');
           check = substr( reverse("&age_2"), 1, 1);
           if check = ',' then call symput ('Total', left(c) );
              else call symput ('Total', left(c+1));
    run;
    %put Total=&total;
```

Figure 3. Step 1 in the Program.

Step 1 in the program finds the number of ranges and/or unique values that have been entered through the prompt. It takes this number and puts it into a macro variable calld **&total.**

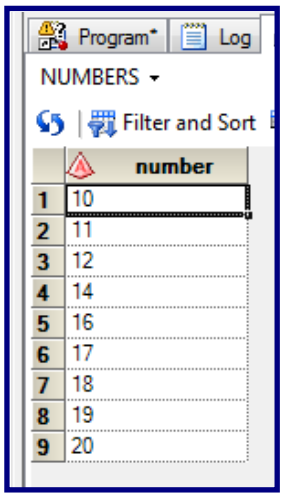
```
*-------------------------------------------------------------------------*;
* Step 2:  Convert Ranges to consecutive numbers.                         *;
*-------------------------------------------------------------------------*;
   data numbers(keep=number );
         array word {&total}  $ 12  word1 - word&total;
         do i = 1 to &total until(word{i} = ' ');
                 word{i} = scan("&age_2", i, ' ,');
                 if index(word{i}, '-') = 0 then do;
                         number =left(word{i}) ;
                         if length(number) = 1  then number='0'!!left(number);
                         if number > 0 then output;
                 end;
                 if index(word{i}, '-') > 0 then do;
                      x=scan(word{i}, 1, '-');
                      y=scan(word{i}, 2, '-');
                      do j=x to y;
                              number=left(j);
                              if length(number) = 1 then number='0'!!left(number);
                              if number > 0 then output;
                      end;
                 end;
         end;
  run;
```

1.

2.

Figure 4. Step 2 in the Program.

The outer loop executes once for each unique value or range. Loop #1 checks to see if there is a dash '-' in the value and if there is NOT one, then we are dealing with a unique value and it is output to the NUMBERS dataset. Loop #2 looks to see if there is a dash and if there is, then we are dealing with a range. This loop writes out every unique value represented by the range to the NUMBERS dataset. At the end of Step 2, the NUMBERS dataset looks like this:



	number
1	10
2	11
3	12
4	14
5	16
6	17
7	18
8	19
9	20

Figure 5. The NUMBERS dataset

The NUMBERS dataset has all the numbers from 10 – 20 except for number 13 and 15. This dataset will be used in the next step to build a WHERE statement that looks like this…

```
WHERE AGE in(10, 11,12, 14, 16, 17, 18, 19, 20);
```

Notice that the variable NUMBER is character.

Step 3 in the program builds a comma separated list of values.

```
*-------------------------------------------------------------------*;
* Step 3: Convert Numbers to a comma separated list of AGE values. *;
*-------------------------------------------------------------------*;
  proc sql noprint;
       select number into : list  separated by ', '
       from numbers;
  quit;
  %put &list;
```

Figure 6. Step 3 in the program.

Viewing the Log shows the results of the %PUT statement.

```
76         !                    %put &list;

10, 11, 12, 14, 16, 17, 18, 19, 20
77
```

Figure 7. The SAS/Log.

Step 4 in the program uses the **&list** variable in a WHERE statement to filter the data.

```
*-----------------------------------------------------*;
* Step 4:  Use the List to build a Where Statement.   *;
*-----------------------------------------------------*;
 title "List of Ages: &list";
  proc print data=sashelp.class;
       title1  "List of Ages: &list  ";
       where age in(&list) ;
  run;
```

Figure 8. The PROC PRINT step.

Notice the **TITLE** statement.

The output looks like this…

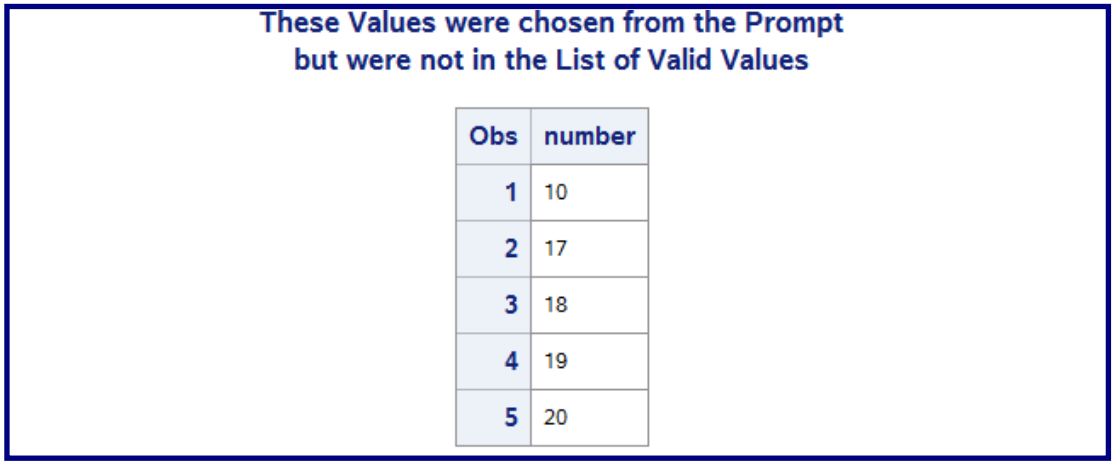
List of Ages: 10, 11, 12, 14, 16, 17, 18, 19, 20

Obs	Name	Sex	Age	Height	Weight
11	Joyce	F	11	51.3	50.5
18	Thomas	M	11	57.5	85.0
7	Jane	F	12	59.8	84.5
13	Louise	F	12	56.3	77.0
6	James	M	12	57.3	83.0
10	John	M	12	59.0	99.5
16	Robert	M	12	64.8	128.0
4	Carol	F	14	62.8	102.5
12	Judy	F	14	64.3	90.0
1	Alfred	M	14	69.0	112.5
5	Henry	M	14	63.5	102.5
15	Philip	M	16	72.0	150.0

Figure 9. Prompt results.

EXAMPLE 2: VERIFYING PROMPT VALUES

The next output takes the results of a verification process and displays values selected in the prompt that are NOT valid. The results of Example 2 are shown below.



These Values were chosen from the Prompt
but were not in the List of Valid Values

Obs	number
1	10
2	17
3	18
4	19
5	20

Figure 10. The results of Example 2.

The next thing we need to do is to create a dataset with the valid values for the prompt. What are the values that are acceptable? We are going to let all the values of AGE in the CLASS dataset be the valid values of AGE. In other words, we are only going to accept values from 11-16 as valid. So, we are going to use PROC FREQ to get the unique values of AGE from the CLASS dataset. And then we are going to use a DATA step to convert these values of AGE to character. The next two steps look like this.

```
*-------------------------------------------------------------------------*;
* Step 5: Get the unique values of AGE to create a list of Valid Values.*;
*-------------------------------------------------------------------------*;
  proc freq data=sashelp.class noprint;
            tables  age / out=freq_out(rename=(age=number));;
  run;

*-------------------------------------------------------------------------*;
* Step 6:  Create a Character Variable from the Numeric Variable.        *;
*-------------------------------------------------------------------------*;
  data freq_out;
       set freq_out(rename=(number=nnumber));
       number = left(put(nnumber, z2.) ) ;
  run;
```

Figure 11. Getting a list of valid values.

The next thing we are going to do is to create a list of INVALID values. Again, these are values entered into the prompt that are not in the acceptable range. This is done in step 7 in the program.

```
*------------------------------------------------------------------------*;
* Step 7: Compare the Values Chosen in the Prompt with a list of Valid  *;
*         Values and generate a Report of NON-Valid values.              *;
*------------------------------------------------------------------------*;
data merged(keep=number);
     merge freq_out(in=valid)
           numbers (in=prompt);
    by number;
    if prompt and not valid;
 run;
proc print data=merged;
      title1  'These Values were chosen from the Prompt';
      title2  "but were not in the List of Valid Values";
     footnote1  "List of Values Selected in Prompt: &list";
 run;
```

Figure 12. Step 7 in the program.

Notice the DATA step. It creates a dataset named MERGED of AGE values that are in entered into the prompt, but are NOT valid. PROC PRINT gives us a report of these values. Notice the titles and footnote.

These Values were chosen from the Prompt
but were not in the List of Valid Values

Obs	number
1	10
2	17
3	18
4	19
5	20

List of Values Selected in Prompt: 10, 11, 12, 14, 16, 17, 18, 19, 20

Figure 13. Invalid Values.

EXAMPLE 3: CREATING AND/OR PROMPT PROMPTS

Suppose you want to combine the effects of **multiple prompts** with **AND/OR** logic? For instance, let say you to add AND/OR logic to the Prompt that we just created. After you select AGE values, you might want to consider examining the values of another variable like HEIGHT. The end result could look something like this…

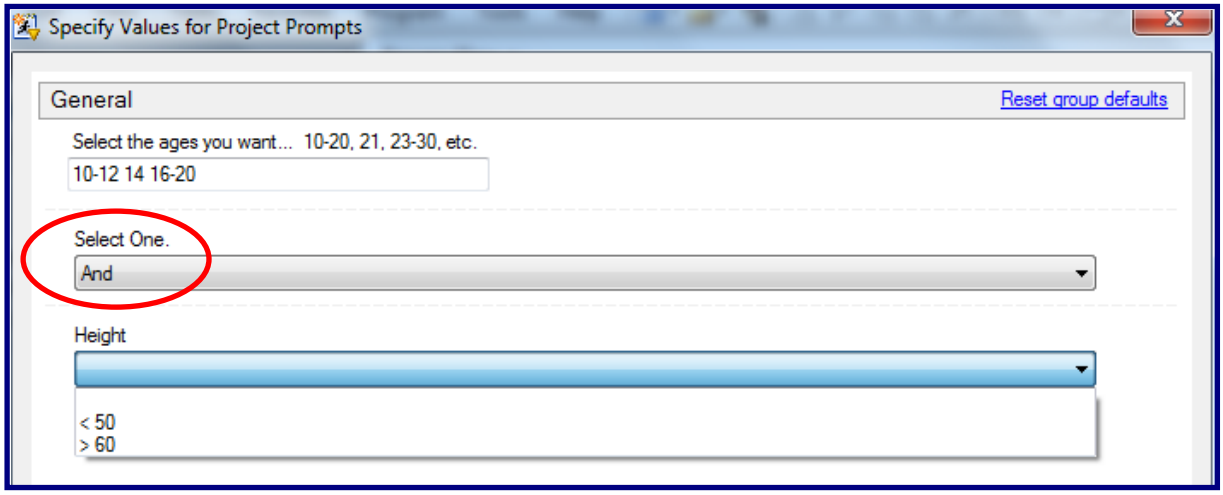


Figure 14.

By selecting **AND** for the second prompt and **> 60** for the HEIGHT prompt, You would get a list of people within the AGE range selected and who were taller than 60 inches.

Go to the Prompt Manager in EG and select Add to start the process of creating a new prompt. Name the prompt something like **And_Or.** Supply something for the displayed text like **'Select One'.**

Next, select the **Prompt Type and Values** Tab and do the following:

1. Set the Prompt Type to **Text.**
2. Make the Number of Values : **Single value.**
3. Make the Method for populating prompt: **User selects values from a static list.**
4. Select **Add,** then manually type in **And.** Select **Add** again and tyupe in **Or.**
5. Select **Ok.**

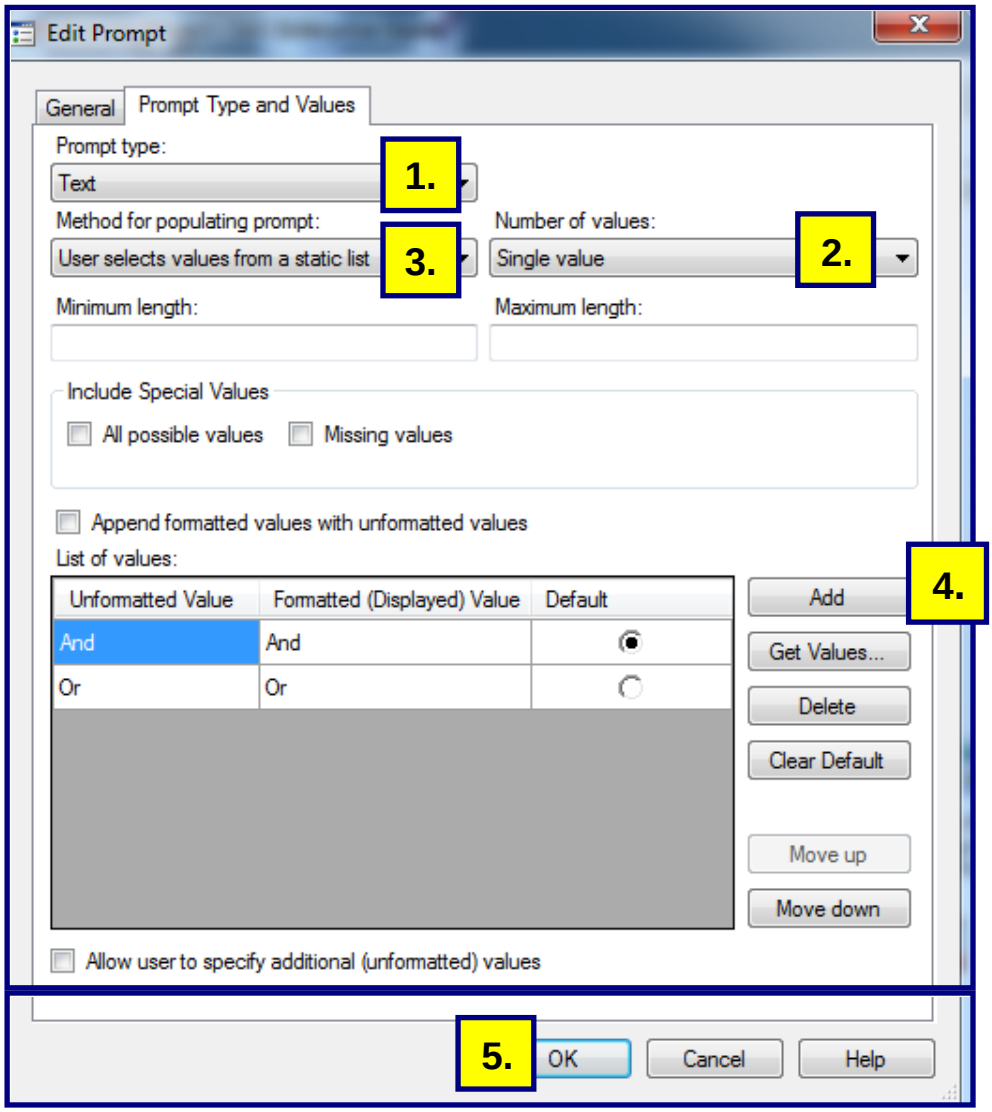


Figure 15.

The next step is to create another prompt and call it **Height.** The General tab should look like this…

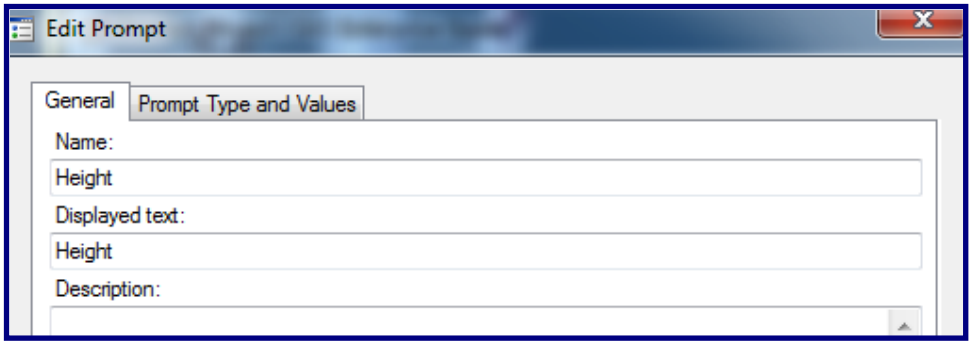


Figure 16.

Select the **Prompt Type and Values** tab and do the following:

1. Set the Prompt Type to **Text.**

2. Make the Number of Values : **Single value.**
3. Make the Method for populating prompt: **User selects values from a static list.**
4. Select **Add,** then manually type in **A <50.** Select **Add** again and type in **>60.**
5. Select **Ok.**

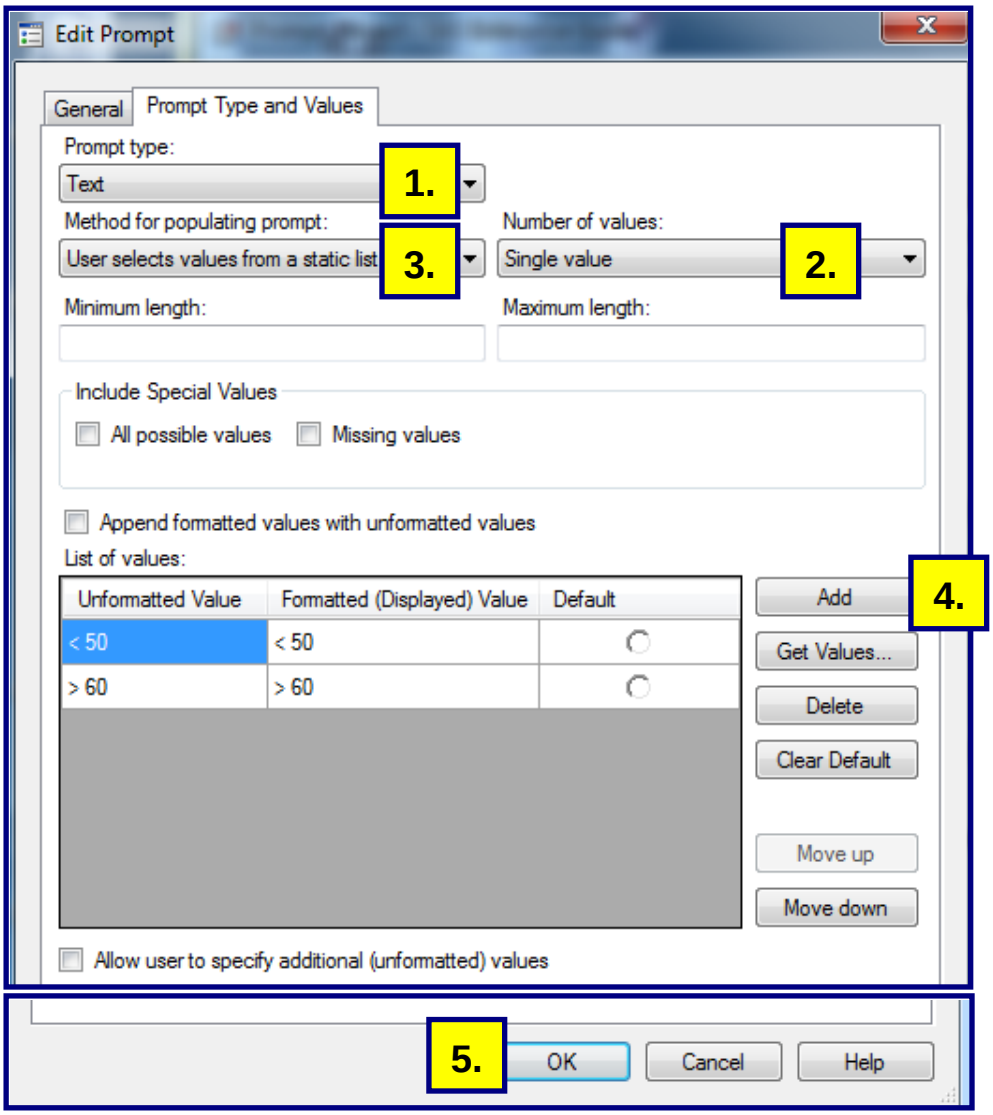


Figure 17.

After the conclusion of the above steps, there are now two new prompts: **And_Or** and **Height.**

Associate these new prompts to the **same program** by doing the following:

1. Go to the Process flow window and find the program icon**.**
2. Right click on it and select Properties**.**
3. Select Prompts from the panel on the left side of the window**.**
4. Select **Add,** then choose the two new prompts **.**

The Properties window for the program should look something like this…

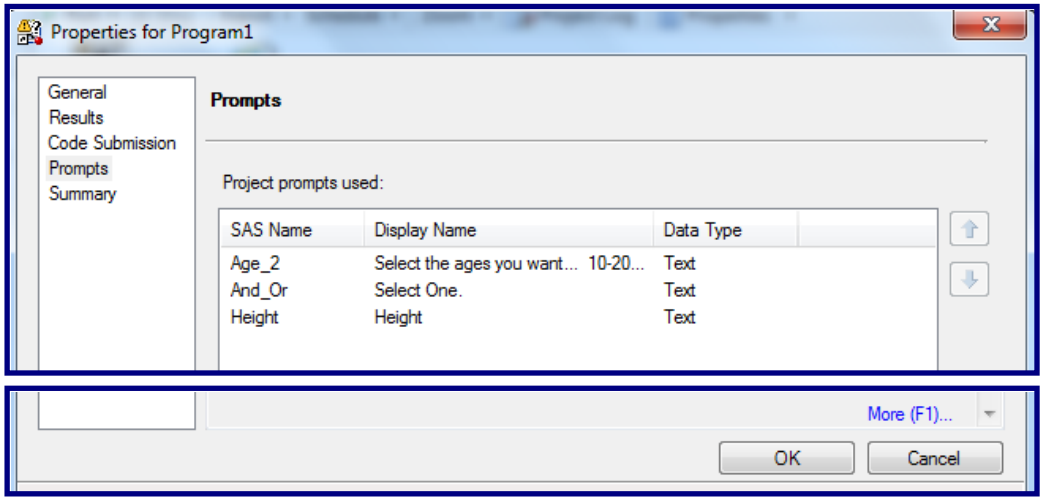


Figure 18. The Properties window.

Select **Ok**, then **Ok** again to go back to the Process Flow window.

Next open the program and edit it by adding the following program segment. The best place to insert these new steps is right after "Step 3" in the existing program.

```
*-------------------------------------------------------------*
* Step 3b.  Augment WHERE statement.                          *
*-------------------------------------------------------------*
 data  _null_;
     %let  where_B= ;
    length where_B $ 300;
    if "&And_or" ne '   ' and "&Height" ne ' '
       then Where_B = "&And_or Height &Height";
          else Where_B = '    ';
    call symput ('Where_B' ,   Where_B);  put WHERE_B;
 run;
 %put Where_B = &Where_B;
*-------------------------------------------------------------*;
* Step 4:  Use the List to build a Where Statement.          *;
*-------------------------------------------------------------*;
 title "List of Ages: &list";
 proc print data=sashelp.class;
    title1  "List of Ages: &list  ";
    title2  "AND/OR is &And_Or";
    title3  "Height = &Height";
    where age in(&list) &Where_B;
 run;
```

Figure 19. Step 3B.

If "**And"** is selected in the **And_Or** prompt, and if **"> 60"** is selected in the **Height** prompt, then the macro variable **&Where_B** resoves to**:**

"And Height > 60" ;

If you make the above mentioned selections in the new prompts when the program runs, the output will look similar to this…

List of Ages: 10, 11, 12, 14, 16, 17, 18, 19, 20
AND/OR is And
Height = > 60

Obs	Name	Sex	Age	Height	Weight
16	Robert	M	12	64.8	128.0
4	Carol	F	14	62.8	102.5
12	Judy	F	14	64.3	90.0
1	Alfred	M	14	69.0	112.5
5	Henry	M	14	63.5	102.5
15	Philip	M	16	72.0	150.0

Figure 20. Proc Print output.

EXAMPLE 4: CUT AND PASTE

This section illustrates how you can cut(copy) and paste values from other files, such as Notepad, Word or spreadsheets into a prompt. Those values are then used to build a WHERE statement in a program. In the next example, we are going to copy values out of a spreadsheet.

The spreadsheet contains a list of questionable transactions. We want to see if any of these Policy Numbers match those in our PREMIUMS SAS dataset. Specifically, we want to know if any of the Policy Numbers from the spreadsheet are making their premium payments.

First, we want to copy the first five policy numbers from the spreadsheet.

.

	A	B	C	D	E	F	G
1	Obs	State	Product	Paid	Policy_Number	date_received	dob
2	1	FL	WMC	8.69	20000	15Jan2012	1Feb1917
3	2	FL	WMC	396.00	20001	22Mar2012	25Jun1975
4	3	FL	WMR	45.00	20002	22Jan2012	8Oct1930
5	4	FL	WMR	348.00	20005	29Jan2012	15Apr1932
6	5	FL	WMC	.00	20010	12Feb2012	25Mar1999
7	6	FL	WMR	225.00	20020	30Apr2012	10Apr1932
8	7	FL	WMR	7.11	20000	21Jan2012	29Jun1928

Figure 21. The spreadsheet

Let's take a quick look at how this prompt was created. The following steps were taken:

1. Went to Prompt Manager in EG and selected **Add.**
2. From the **Add a New Prompt** window, selected the **General** tab.
3. Supplied a Name, like **Cut_and_Paste.**
4. Supplied the text to be displayed at runtime.
5. Selected the **Prompt Type and Values** tab.

From the **Prompt Type and Values** tab, the following was done:

1. Made the Prompt type: **Text.**
2. Made the population method: **User enters values.**
3. Made the Number of values: **Single value.**
4. Made the Text type: **Multi-line text.**
5. Selected **Ok.**

Before the Add a New Prompt window closed, it looked like this:

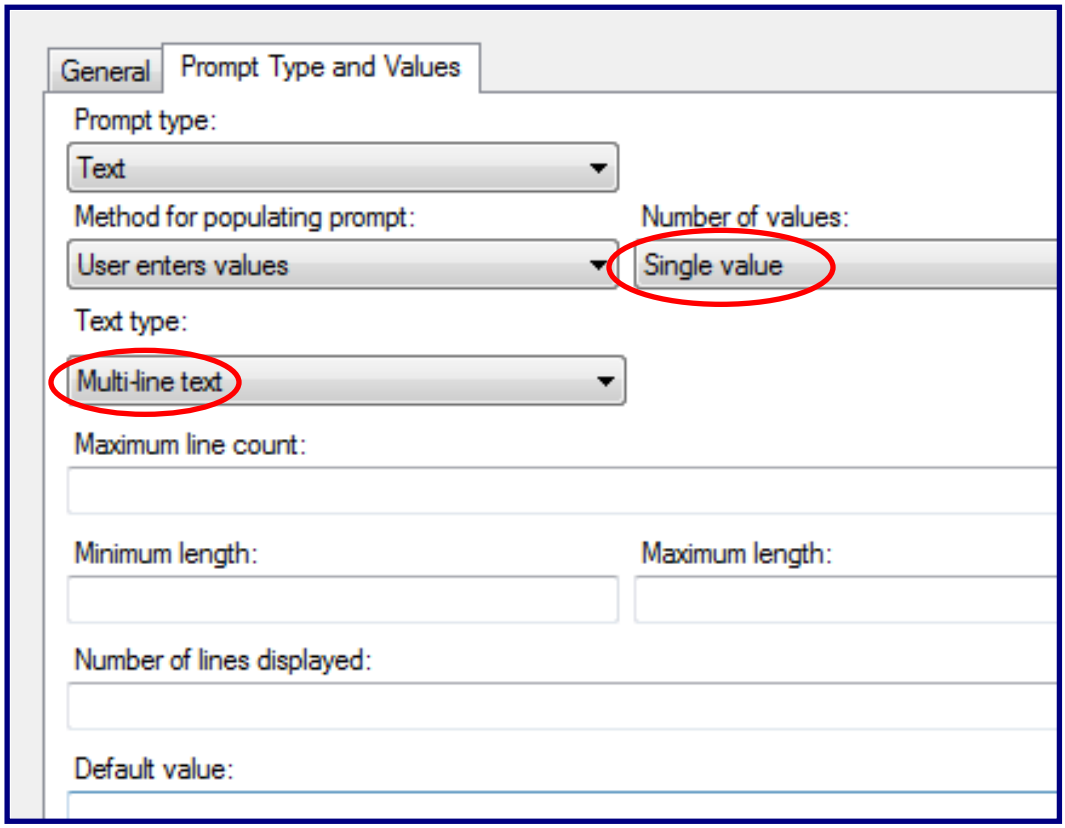


Figure 22.

The two most important things to do with this prompt are circled in in Figure 22.

Before we fully utilize this prompt in a program, it's a good idea to see what this prompt 'delivers' to the program. So, the first thing we are going to do is associate the prompt with a new program and then test the prompt.

Select File → New → program while in EG to create the new program.
Just put this statement in the program editor:

```
%put &Cut_and_Past;
```

Close the program. From the Process Flow window, right click the program icon. Rename the program **"Prompts".** Associate the **Cut_and_Paste** prompt with the program (see steps at the bottom of page 9.).

Run the program and paste the values in the prompt. It should look like this:

Figure 23. The Cut and Paste prompt.

Select **Run.** Then open the Log**.**

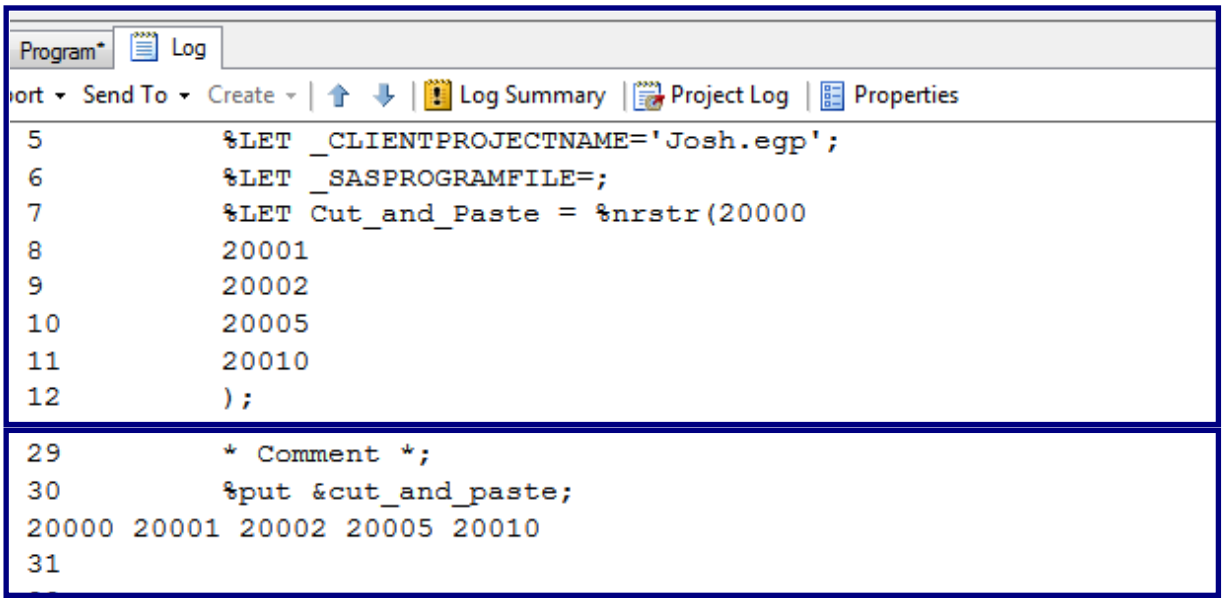


Figure 24. The SAS Log.

Notice the values of &Cut_and_Paste. The next step is to write code that can manipulate these values to filter data. Go back to the program and add this DATA step.

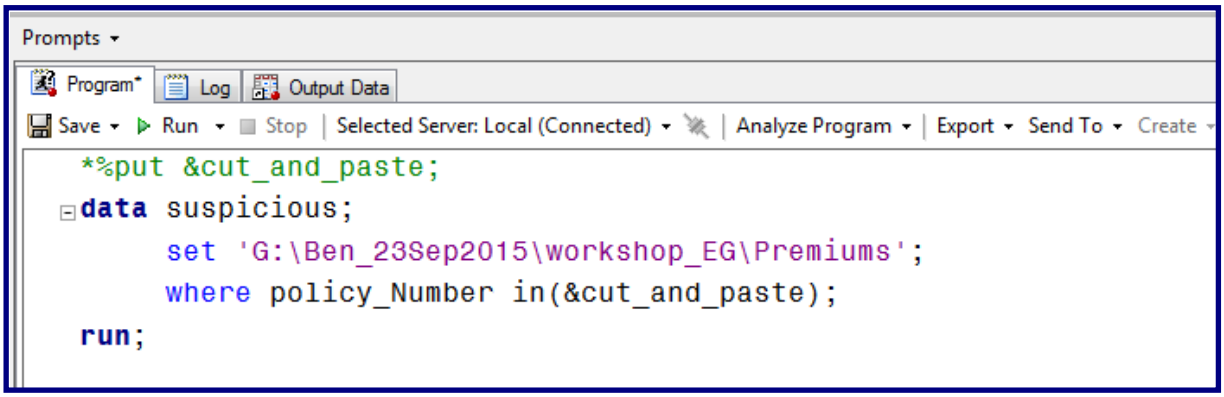


Figure 25. Final DATA Step.

Notice the **WHERE** statement. Add a PROC PRINT step, run the program and paste the values copied from the spreadsheet into the prompt.

Obs	Policy_Number	state	Premium	date_paid	Product	year	county	New_County
1	20000	FL	95.00	09JUN2012	WMR	2005	HILLS	HILLS
2	20000	FL	95.00	19JUL2010	WMR	2003	HILLS	HILLS
3	20000	FL	95.00	30JUL2013	WMC	2006	HILLS	HILLS
4	20000	FL	95.00	13DEC2009	WMR	2003	HILLS	HILLS
5	20000	FL	95.00	28FEB2011	WMR	2004	HILLS	HILLS
6	20000	FL	95.00	17SEP2010	WMC	2003	HILLS	HILLS
7	20000	FL	95.00	31MAY2012	WMR	2005	HILLS	HILLS
8	20000	FL	95.00	09DEC2012	WMC	2006	HILLS	HILLS
9	20000	FL	95.00	29MAY2012	WMR	2005	HILLS	HILLS

Figure 26. PROC PRINT output.

CONCLUSION

Prompts are fairly simple to create, but have very powerful effects on your SAS programs. Prompts can also be used in the Query Builder in Enterprise Guide. The real power of prompts can be harnessed when their results are incorporated into programs. It is the hope of the author that the reader of this paper has learned some valuable information about how to create and use prompts.

ACKNOWLEDGMENTS

I would like to thank my clients for this past year in posing prompt challenges to me. These challenges have been the source of inspiration for this presentation. I thank them also for their patience because I did not always give them a prompt solution.

CONTACT INFORMATION

If you have any questions or comments, the author can be reached at:

Ben Cochran
The Bedford Group
3224 Bedford Avenue
Raleigh, NC 27607
Work Phone: 919.741.0370
Email: bencochran@nc.rr.com