Paper 818-2017

Getting Started with SAS® Prompts

Brian Varney, Experis Business Analytics Practice

ABSTRACT

Allowing SAS® users to leverage SAS prompts when running programs is a very powerful tool. Using SAS prompts makes it easier for SAS users to submit parameter driven programs and for developers to create robust, data driven programs. This presentation will demonstrate how to create SAS prompts from SAS Enterprise Guide and roll them out to users so they can take advantage of them from SAS programs and SAS Stored Processes.

INTRODUCTION

SAS prompts allow developers to provide programs to users such that they can change parameter values when they run a program without touching the SAS code itself. The developer can leverage the attributes of the prompt to define constraints on the values entered or chosen.

There are many advantages to using prompts when rolling out programs to other users and consumers.

1) SAS programs can be rolled out to report consumers without providing access to the program code. The person running the report does not alter to code to change parameter values. The source code is protected.

2) SAS prompts are presented to users in a nice graphical interface which is more user friendly than manually changing values in a program.

3) Constraints can be placed on the values chosen or entered into a prompt to minimize the risk of erroneous values being passed into programs.

4) The interface for choosing parameter values is more professional looking than having to go in and altering code.

5) A stored process can easily be created such that the prompt driven program can be executed from many different SAS client desktop applications such as SAS Enterprise Guide, SAS Add-in for Microsoft Office, SAS Stored Process Web Application, SAS Information Delivery Portal.

The remainder of this paper will introduce SAS prompts, explain how they work, and show examples of the macro variables generated in the background. The scope of this paper will be limited to creating and using SAS Prompts with SAS parameter driven code from SAS Enterprise Guide.

LAYING THE FOUNDATION FOR WHAT THIS PAPER ADDRESSES

We have all seen parameter driven programs similar to the code below:

```
%macro doit(study=, num_obs=);

title1 "Contents for demo_&study.";
proc contents data=demo_&study.;
run;

options obs=&num_obs.;
title1 "Listing for demo_&study.";
proc print data=demo_&study.;
run;
options obs=max;

%mend doit;

%doit(study=P1001, num_obs=5);
```

The code stays static except for the study number and number of records to display in the listing. In some environments, programs such as this would have to be opened by the report runner so they could alter the program code or macro call. Another scenario is that every time someone wants the report run, they contact the report developer and requests them to run it. By using SAS Prompts and SAS Stored Processes, the code can be kept static and the report developer does not need to be bothered with report requests.

A STRATEGY FOR DEVELOPING A PROGRAM WITH SAS PROMPTS

PLANNING

There are a few aspects to plan when creating prompts to use with a parameter driven program.

- Do you want the user to enter in text or have to choose values from a list?
- What values are possible for the prompt to take on?
- Are there constraints or rules that I want enforced for the values entered or chosen?
- What do you want the report runner to see?
- Will the prompts depend on the value chosen by another prompt?
- How can I minimize the risk of error or confusion?

When creating SAS prompts to be used with a SAS program and/or SAS Stored Process, I like to keep track of what is happening behind the scenes from a SAS Macro variable standpoint. While creating and developing the prompts, I will create a one line program to attach the prompts to as shown below.

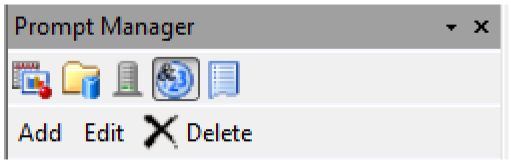
```
%put _all_;
```

What the above code does is show me all of the SAS macro variables. Since SAS Prompts create SAS Macro variables and populate them with values, this is the perfect way to understand what the prompt is creating in the background. This makes it much easier to modify or create the SAS code to take advantage of the SAS Macro variables in the background.

EXAMPLE – CREATING A SAS PROMPT

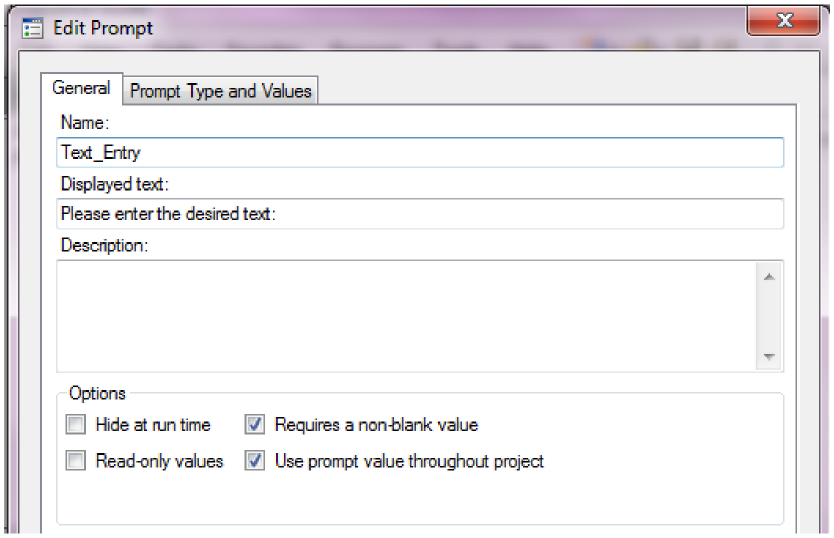
Let's look at a simple example. This example allows the user to enter in their first name which will be available to be used within the SAS program as a macro variable value.

Assuming we have SAS Enterprise Guide open, navigate to the Prompt Manager. In the middle left of SAS Enterprise Guide you will see the following 5 icons. The Prompt Manager is the fourth one from the left.



If you click on the "Add" button, an interface will appear that will allow you to create a SAS Prompt. There are two tabs: "General" and "Prompt Type and Values".

Let's begin with the "General" tab.



Name:
Text_Entry
Displayed text:
Please enter the desired text:
Description:

Options
☐ Hide at run time ✓ Requires a non-blank value
☐ Read-only values ✓ Use prompt value throughout project

Name

This will be the name of your prompt. It needs to be a valid SAS Macro variable name. In this example, the macro variable created and available to the SAS program that this prompt is attached to is text_entry. To reference it in a program, we would use &text_entry.

Displayed Text

The displayed text is what the user sees when the prompt is presented to them.

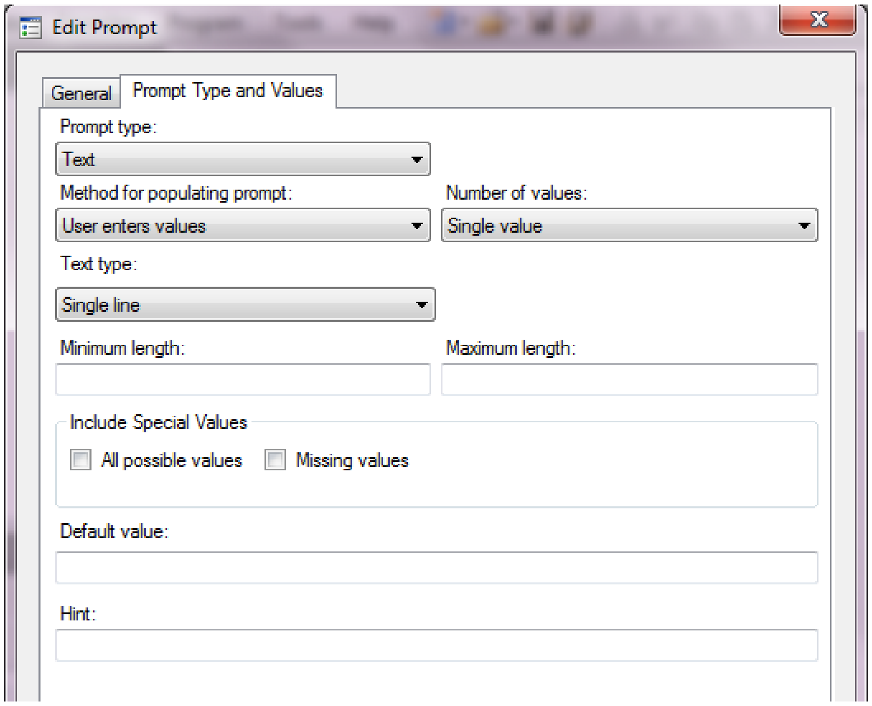
Description

A description can be entered about the prompt. This is optional.

Options

These options control some of the prompt behaviors. For the SAS Prompts I have used in my work and for the sake of this paper, we will have the two check boxes on the right checked.

Now let's focus on the "Prompt Type and Values" tab.


Edit Prompt

General | Prompt Type and Values

Prompt type: Text

Method for populating prompt: User enters values

Number of values: Single value

Text type: Single line

Minimum length:

Maximum length:

Include Special Values

[ ] All possible values [ ] Missing values

Default value:

Hint:

Prompt Type

There are many different prompt types. I will not list them all here but I would encourage you to investigate the prompt types that we do not cover examples on in the paper and experiment with them.

Please note that depending on the type of prompt that you select, the rest of the options may change to cater to what is relevant for that type of prompt.

For this example, we will just have a simple Text prompt that a user enters one value into.

Method for Populating Prompt

In this example, we are going to have the user enter in a value. But we could also have them select one or more values from a *static* or *dynamic list*.

A *static list* is a list of values that is entered during the prompt creation or loaded in once from a SAS data set variable.

A *dynamic list* is a list of values from a SAS data set registered in the SAS metadata. If the variable values linked to the prompt change, so do the values presented to the user when running the SAS prompt.

Number of Values

You can have the user enter in multiple lines of text.

Text Type

You can specify the text entry to be masked and passed as an encoded or non-encoded string for something like a password.

Minimum/Maximum Length

Define a constraint for a minimum or maximum length for the text the user enters.

Include Special Values

For prompts that allow the user to select more than one value, you can have special values of missing and/or “All Selected Values” included. This is not relevant for prompts in which the user enters in text; only ones that they select values from a list.

Default Value

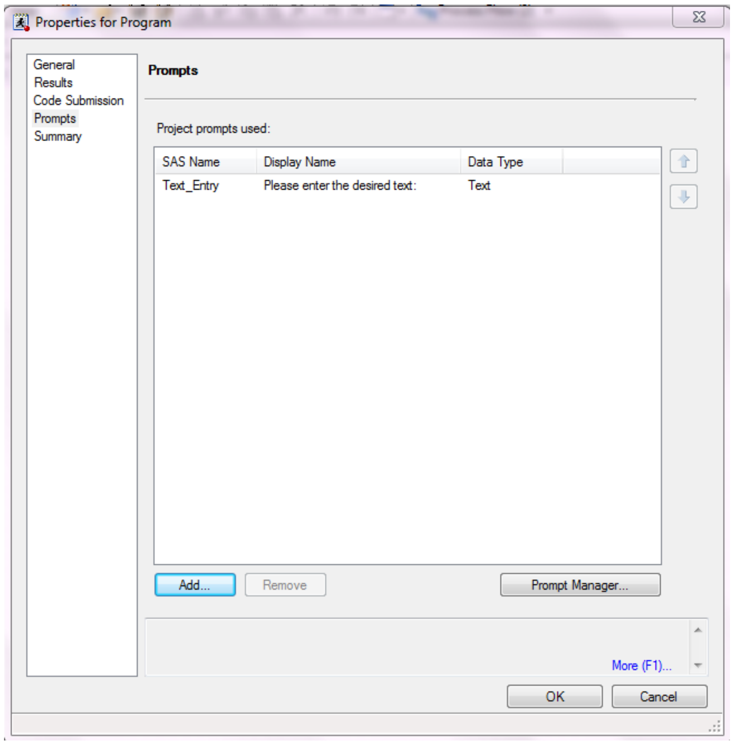
Creates a default value for the prompt and this is presented when the prompt is displayed.

Hint

A hint presented to the user that is intended to clear up any confusion. The hint is presented below the text prompt.

EXAMPLE – LEVERAGING A SAS PROMPT WITH A SINGLE VALUE FROM A SAS PROGRAM

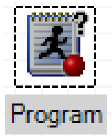
If you have a SAS program in SAS EG, you can right click on it and choose properties. One of the properties is “Prompts”. You can add as many SAS prompts as you need for your parameter driven program. As you can see below, we chose the SAS Prompt we created called text_prompt.



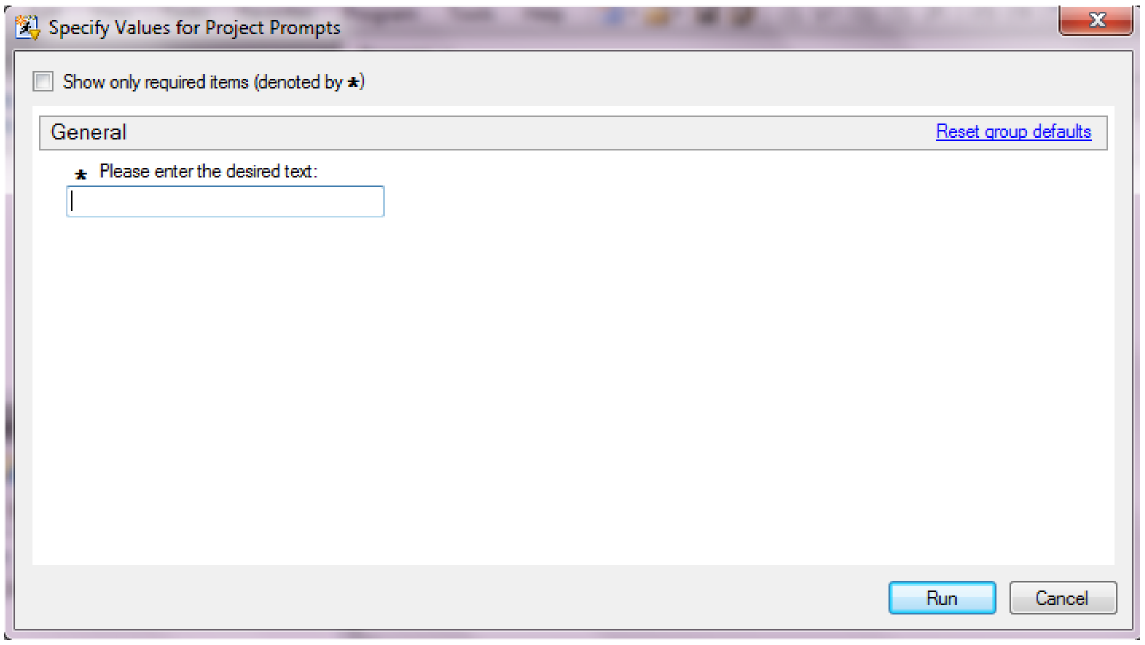
Project prompts used:

SAS Name	Display Name	Data Type
Text_Entry	Please enter the desired text:	Text

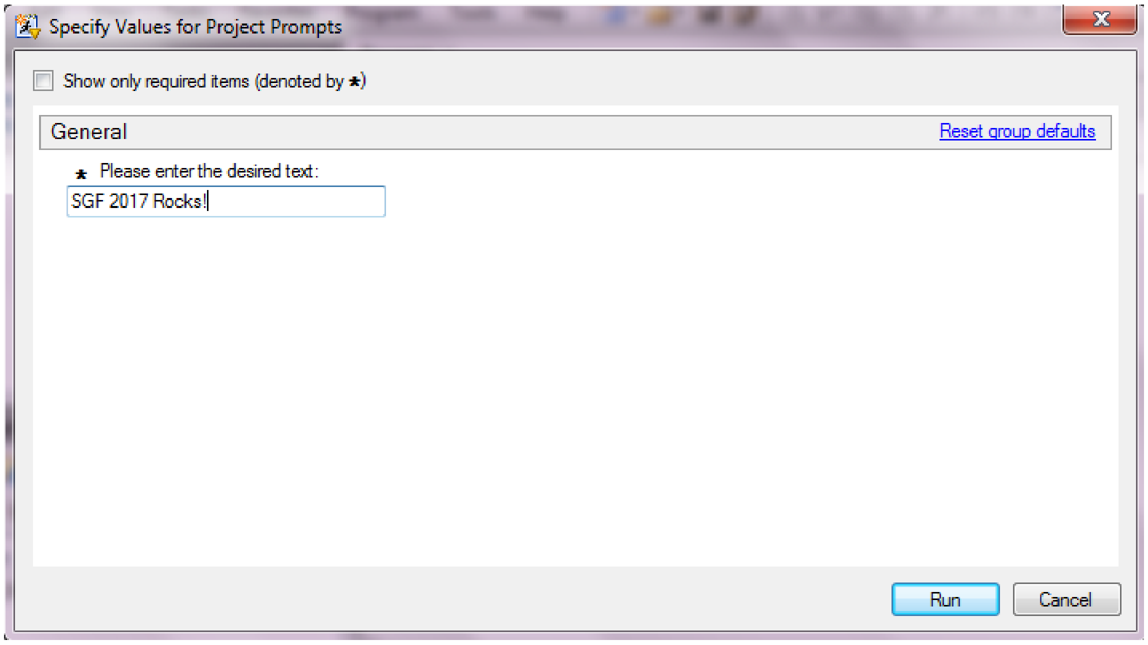
After you add the prompts and click OK, you will notice that he SAS program in your SAS EG project has a question mark on it.



If you remember, we defined a simple SAS Prompt asking the user to enter one text value. The following dialogue is presented to the user.


Specify Values for Project Prompts
Show only required items (denoted by ⁎)
General
Reset group defaults
⁎ Please enter the desired text:
Run
Cancel

Now let’s enter text into the prompt and see what happens behind the scenes.


Specify Values for Project Prompts

Show only required items (denoted by ⁎)

General

⁎ Please enter the desired text:

SGF 2017 Rocks!

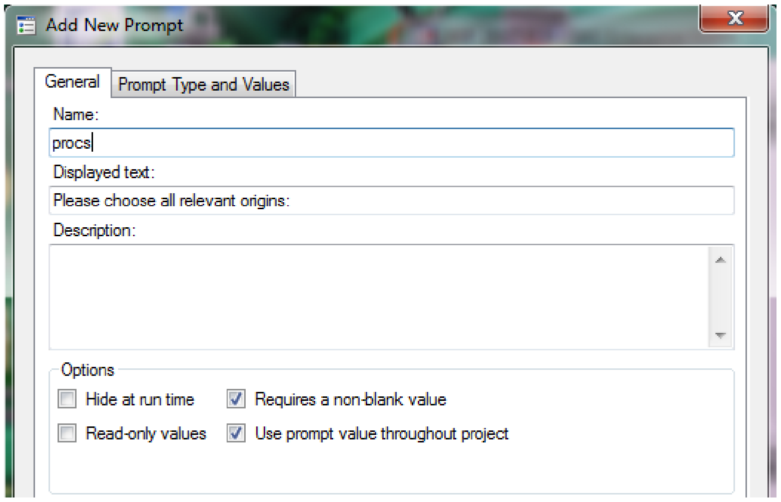
After clicking on the “Run” button at the bottom, we look in the log and see the following code generated by this prompt.

```
%LET Text_Entry = SGF 2017 Rocks!;
```

Now we have the global macro variable text_entry available to us to use in our program.

EXAMPLE – LEVERAGING A SAS PROMPT WITH MULTIPLE VALUES FROM A SAS PROGRAM

Now let’s create a second prompt called procs.


Add New Prompt

Name: procs

Displayed text: Please choose all relevant origins:

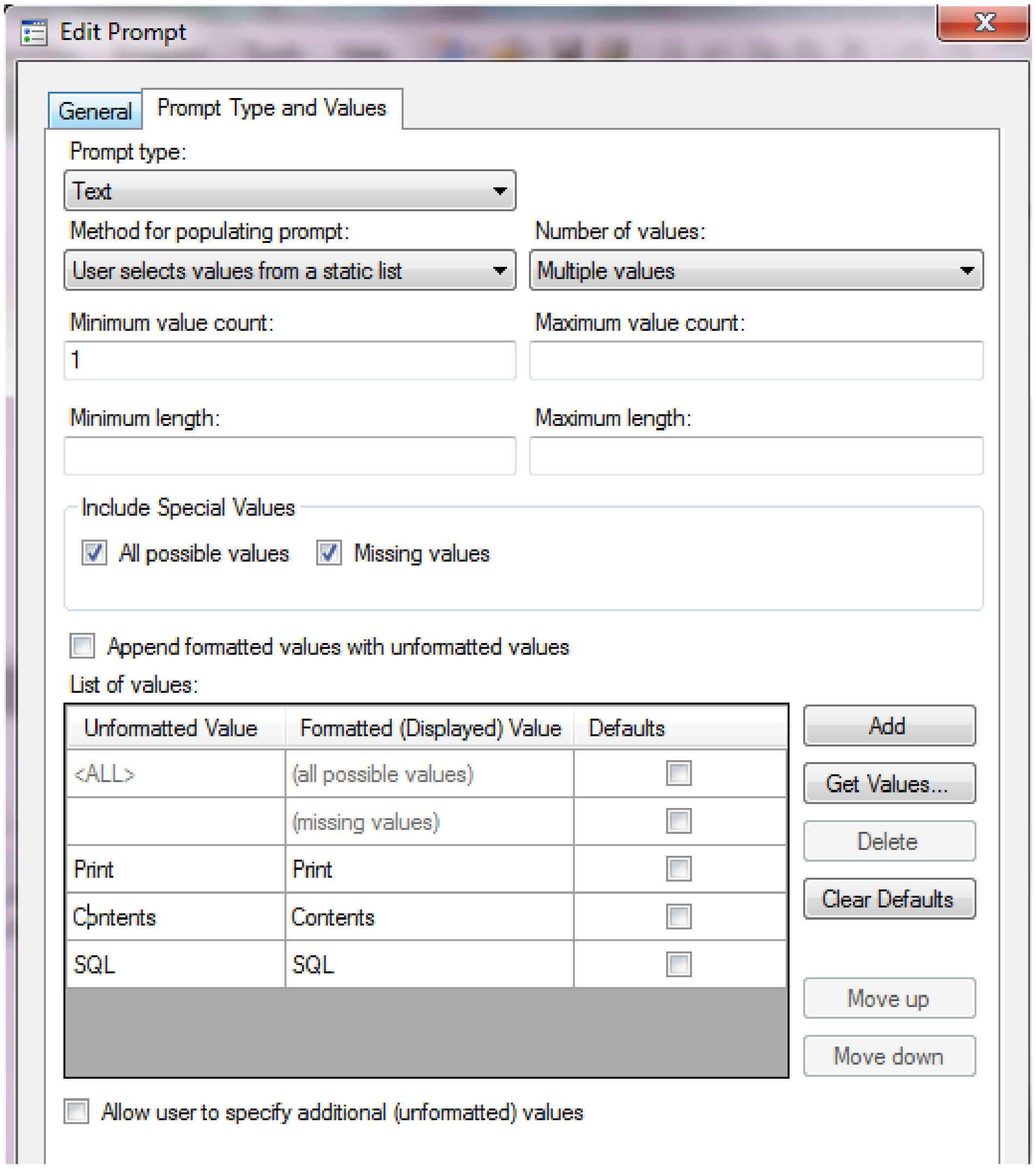
Description:

Options

☐ Hide at run time ☑ Requires a non-blank value

☐ Read-only values ☑ Use prompt value throughout project

This prompt is still a text prompt but the user can choose multiple values. In this example we entered values into a static list manually using the “Add” button. We also chose to “Include Special Values” for the user to choose; one for “All possible values” and the other for missing values.



Prompt type: Text

Method for populating prompt: User selects values from a static list

Number of values: Multiple values

Minimum value count: 1

Maximum value count:

Minimum length:

Maximum length:

Include Special Values

[✓] All possible values [✓] Missing values

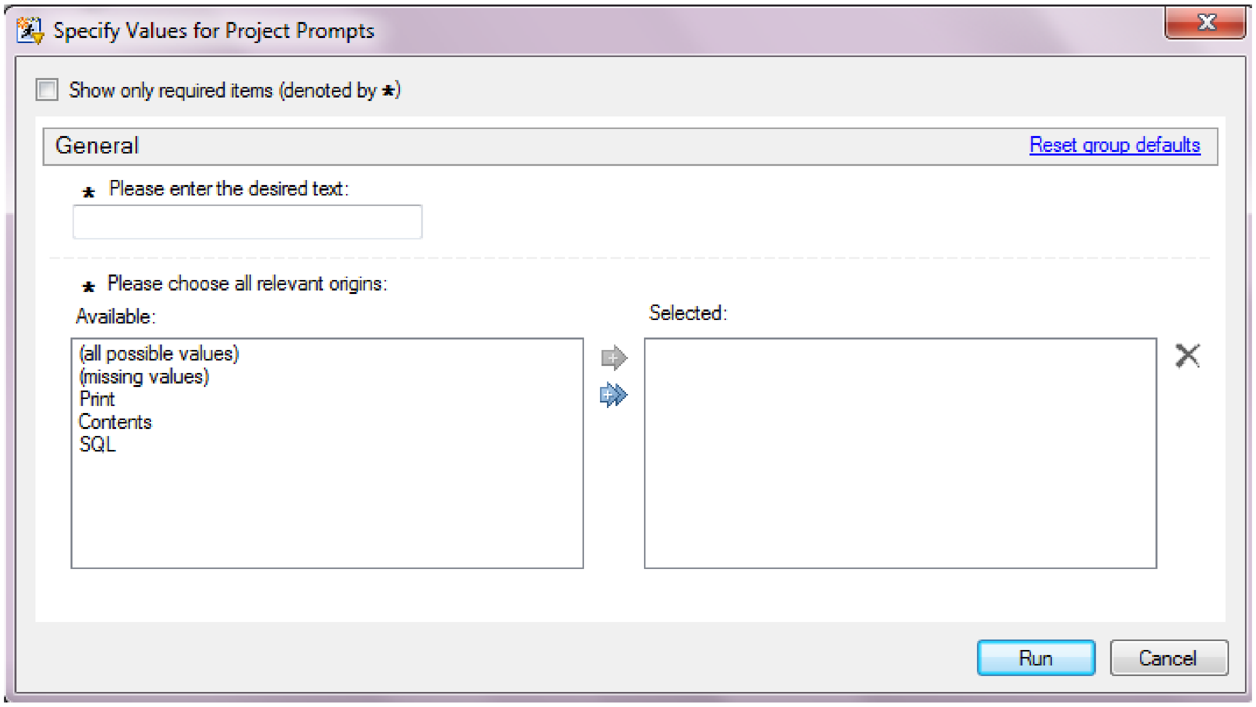
[ ] Append formatted values with unformatted values

List of values:

Unformatted Value	Formatted (Displayed) Value	Defaults
<ALL>	(all possible values)	☐
	(missing values)	☐
Print	Print	☐
Contents	Contents	☐
SQL	SQL	☐

[ ] Allow user to specify additional (unformatted) values

Since we added the second prompt in the properties for the program, the prompt window presented to the user now has both the first prompt we created along with the second prompt.


Specify Values for Project Prompts
Show only required items (denoted by ✱)
General
Reset group defaults
Run
Cancel

✱ Please enter the desired text:

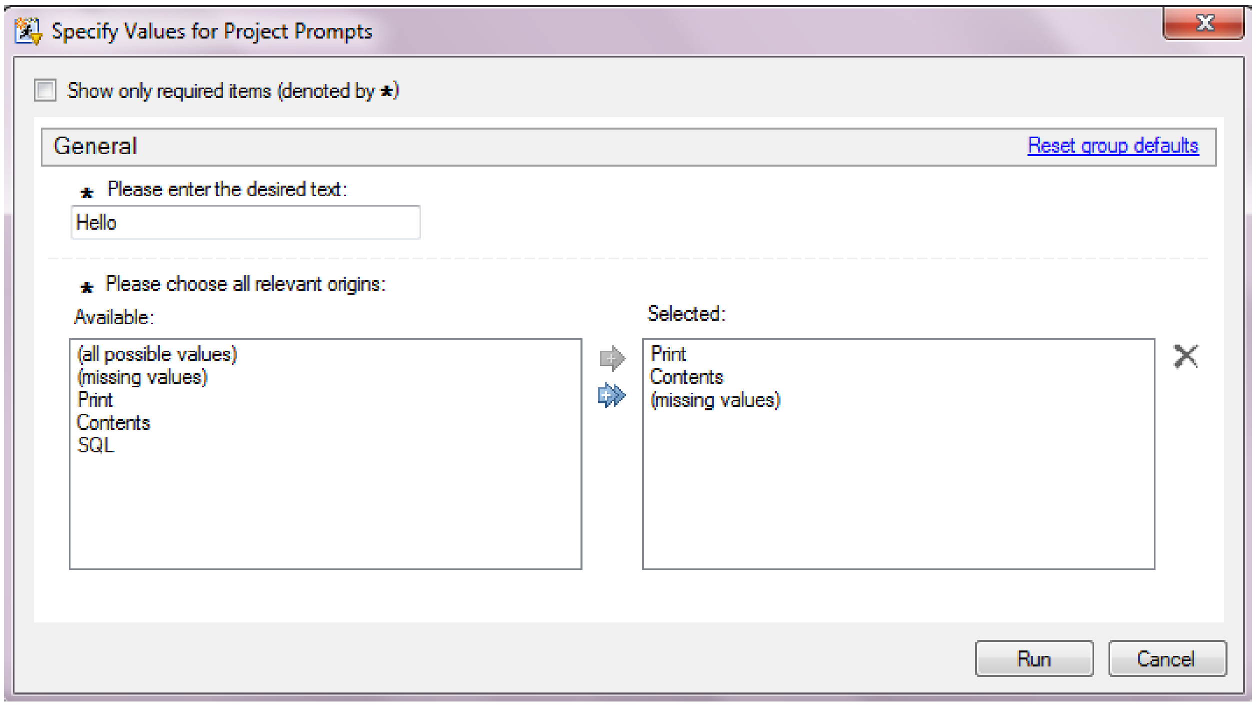
✱ Please choose all relevant origins:

Available:

(all possible values)
(missing values)
Print
Contents
SQL

Selected:

We type in a value for the first prompt and then choose three values in the second prompt: “Print”, “Contents”, and “(missing values)”.


Specify Values for Project Prompts
Show only required items (denoted by ✱)
General
Reset group defaults
Run
Cancel

✱ Please enter the desired text:

Hello

✱ Please choose all relevant origins:

Available:

(all possible values)
(missing values)
Print
Contents
SQL

Selected:

Print
Contents
(missing values)

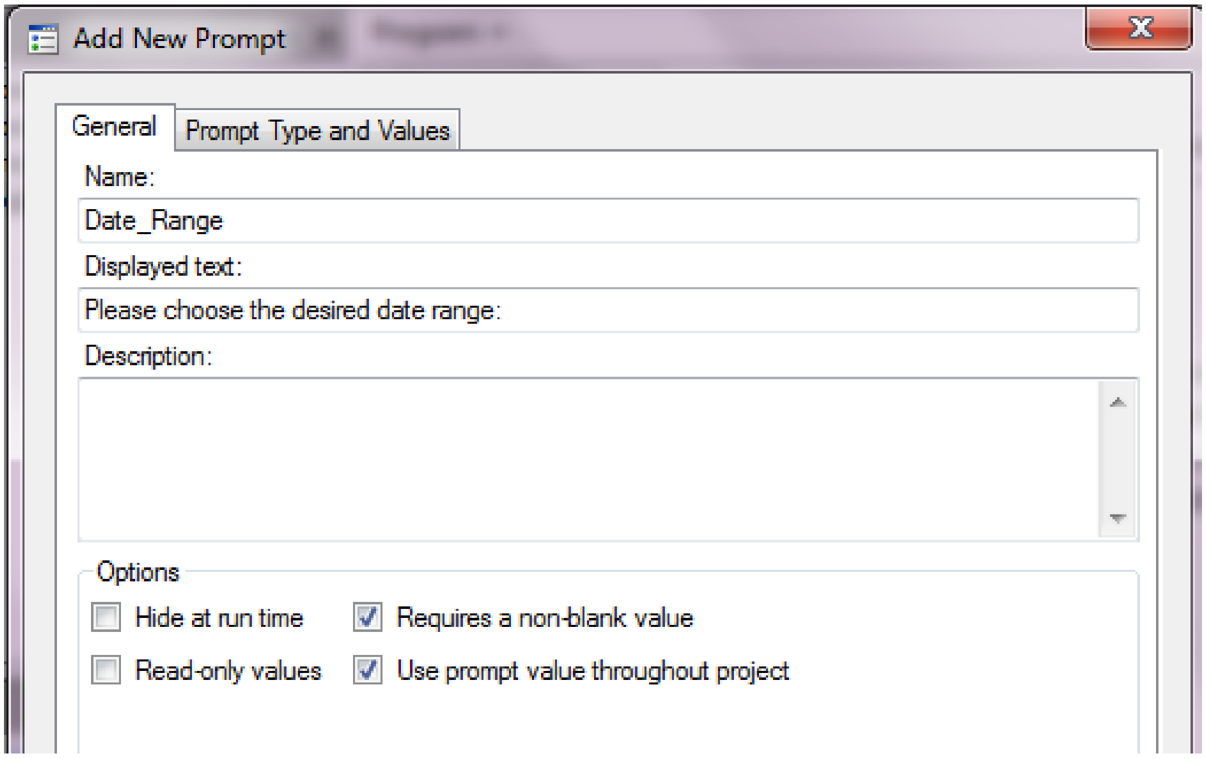
After clicking run, the two prompts, text_entry and procs, generate the following %let statements based on the values entered and chosen in the prompt window.

```
%LET Text_Entry = Hello;

%LET procs0 = 3;
%LET procs3 =  ;
%LET procs2 = Contents;
%LET procs_count = 3;
%LET procs = Print;
%LET procs1 = Print;
```

EXAMPLE – LEVERAGING A DATE RANGE SAS PROMPT FROM A SAS PROGRAM

For our last example, we will define a date range prompt.



Name: Date_Range

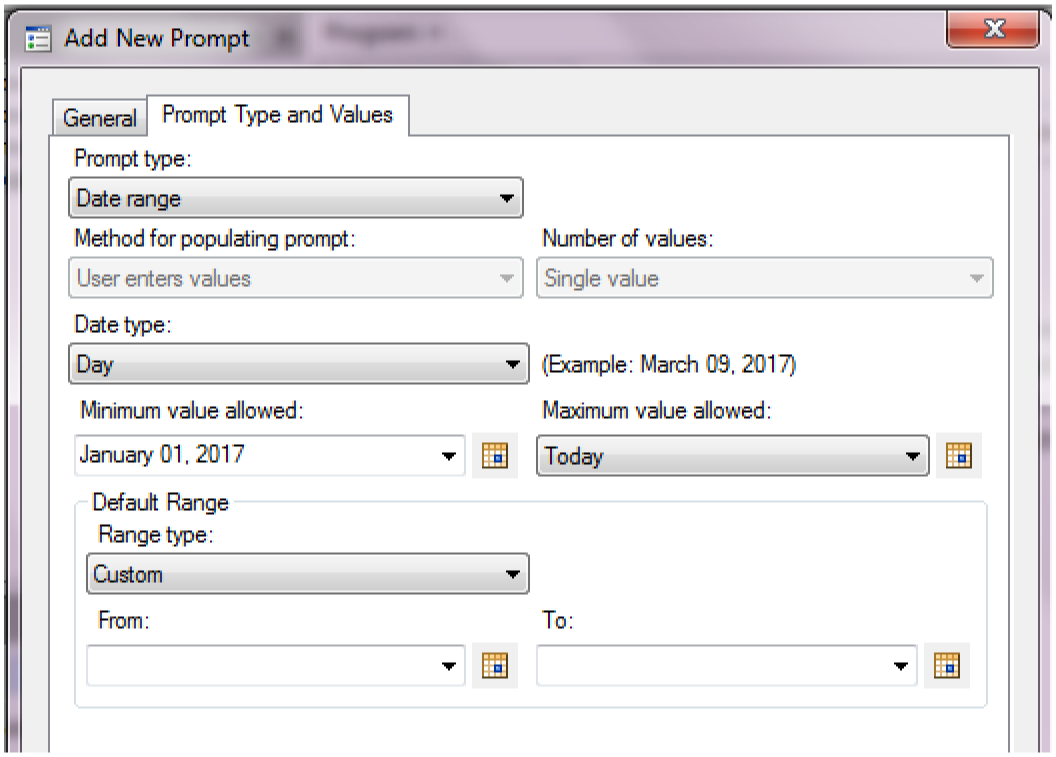
Displayed text: Please choose the desired date range:

Description:

Options

[ ] Hide at run time
[✓] Requires a non-blank value
[ ] Read-only values
[✓] Use prompt value throughout project

This date range prompt is going to return day values (ddmonyyyy). As you can see, we define a minimum as a value of January 1st, 2017. The maximum value is relative based on the date we are running the SAS program.


Add New Prompt

General | Prompt Type and Values

Prompt type: Date range

Method for populating prompt: User enters values

Number of values: Single value

Date type: Day (Example: March 09, 2017)

Minimum value allowed: January 01, 2017

Maximum value allowed: Today

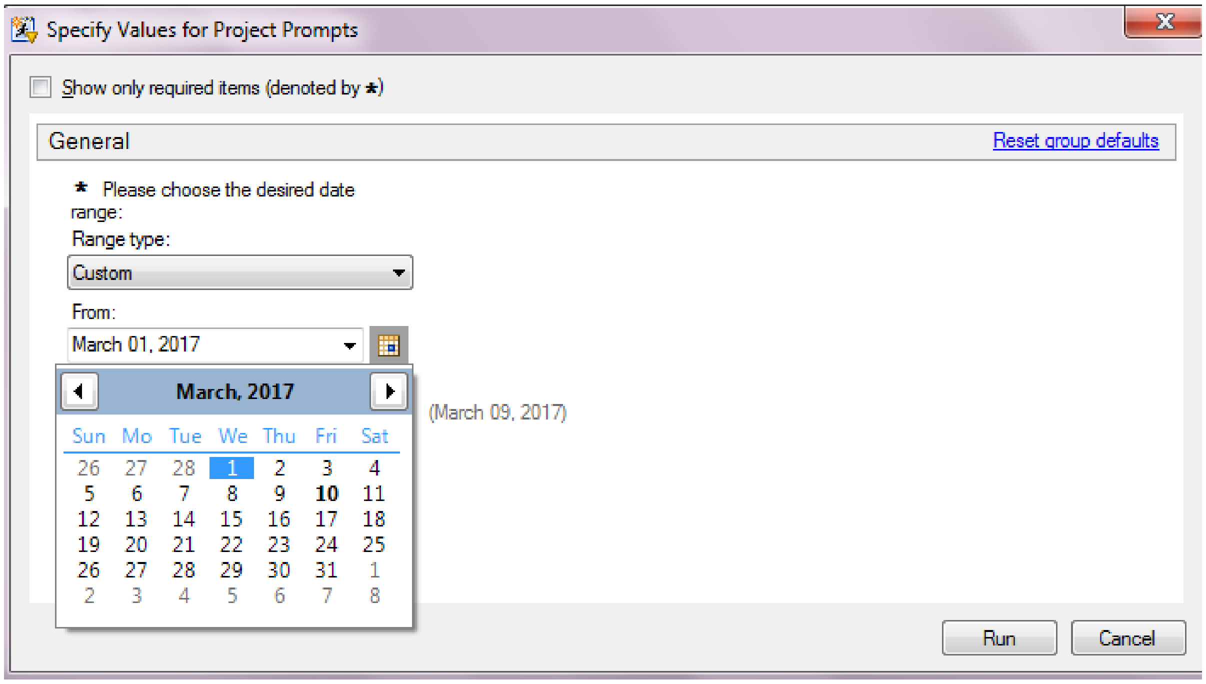
Default Range

Range type: Custom

From:

To:

We went into the properties of the program, took off the first two prompts we were working with and added our date_range prompt. When we run the program, we can click on the little calendar icon and choose the date values from a calendar.


Specify Values for Project Prompts

Show only required items (denoted by *)

General — Reset group defaults

* Please choose the desired date range:

Range type: Custom

From: March 01, 2017

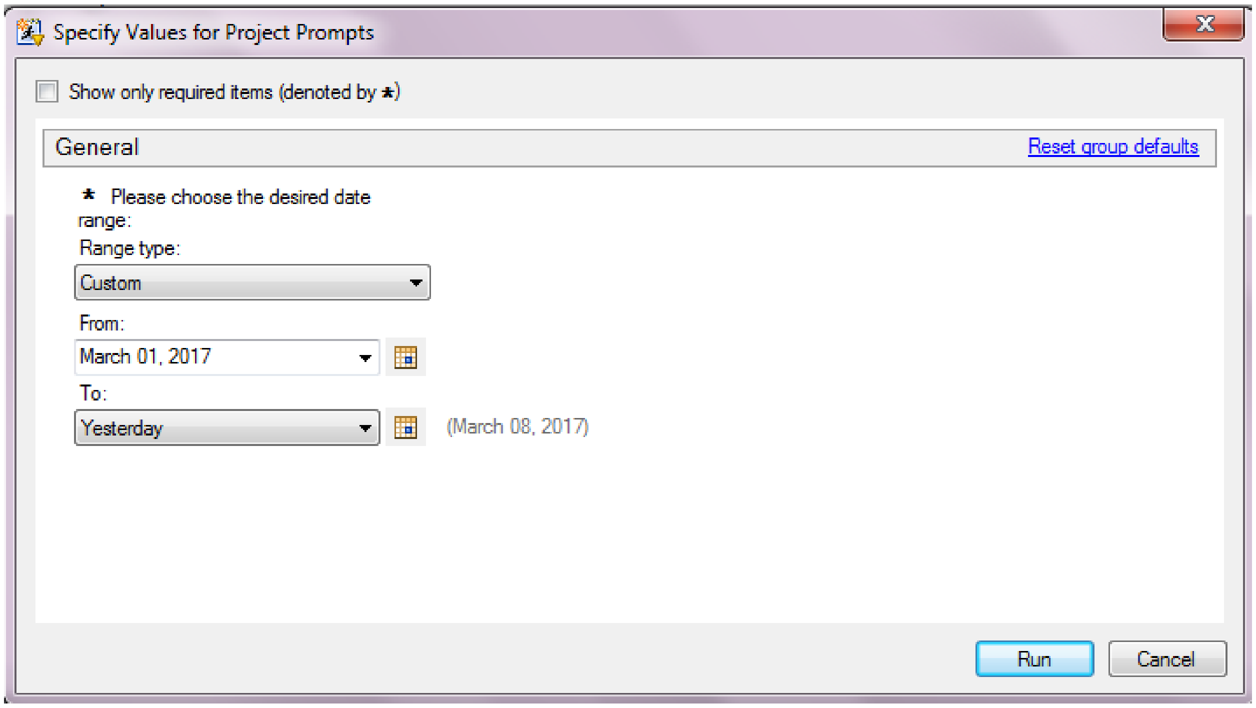
(March 09, 2017)

March, 2017

Sun	Mo	Tue	We	Thu	Fri	Sat
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Run | Cancel

We have chosen our date range values and are ready to click run.


Specify Values for Project Prompts
Show only required items (denoted by *)
General
Reset group defaults
Run
Cancel

* Please choose the desired date range:
Range type: Custom
From: March 01, 2017
To: Yesterday (March 08, 2017)

After clicking “Run”, the date_range prompt generates the following %let statements.

```
%LET Date_Range_max = 08Mar2017;
%LET Date_Range_max_rel = D-1D;
%LET Date_Range_min_label = March 01, 2017;
%LET Date_Range_max_label = Yesterday;
%LET Date_Range_min = 01Mar2017;
```

CONCLUSION

Many companies bring in server based SAS environments but never fully utilize the functionality regarding SAS Stored Processes and SAS Prompts. Taking parameter driven programs and adding SAS prompts will create much more secure, consistent, and robust experiences for SAS developers and report consumers.

CONTACT INFORMATION

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

Brian Varney
Experis Business Analytics Practice
(269) 365-1755
brian.varney@experis.com
www.experis.us/analytics
www.experis.us/clinical