Paper 1245-2014

Uncover the Most Common SAS® Stored Process Errors

Tricia Aanderud, And Data Inc

Angela Hall, SAS Institute

ABSTRACT

You don't have to be with the CIA to discover why your SAS® stored process is producing clandestine results. In this talk, you will learn how to use prompts to get the results you want, work with the metadata to ensure correct results, and even pick up simple coding tricks to improve performance. You will walk away with a new decoder ring that allows you to discover the secrets of the SAS logs!

INTRODUCTION

There are few things more ego deflating than running your beautiful stored process only to see it fail. Either the stored process complains it cannot complete the process or even worse – no results at all! If you are used to writing SAS programs that only you run, it is a little easier to debug issues or instinctively know how to avoid them. If you have never written a stored process, refer to *Creating Your First Stored Process* in the Reference section.

This paper explains some ways to prevent issues, how to find issues when they do occur, and some help with ensuring a speedy stored process.

PREVENTING ISSUES

Obviously, the best way to ensure you stored process works is to prevent the user from experiencing any issues. When using prompts there are some easy ways to help the user make the right selections.

DECODER TRICK #1: REQUIRE AN ANSWER

If your stored process code will break unless it gets a prompt value, require the user to provide a value. If your code is similar to the following example, it will fail if the user does not provide the &StatusPrompt value:

```
proc print data=OPS.orders;
where status = "&StatusPrompt";
run;
```

If the stored process requires a prompt value – problem solved. When you create the stored process, select the **Requires a non-blank value** check box on the General pane. With this selection, SAS will not let the user move forward without an answer. SAS adds an asterisk to the prompt and if the user selects Run without providing an answer, an error message is generated. Very nice built-in functionality that will save you hours of coding.

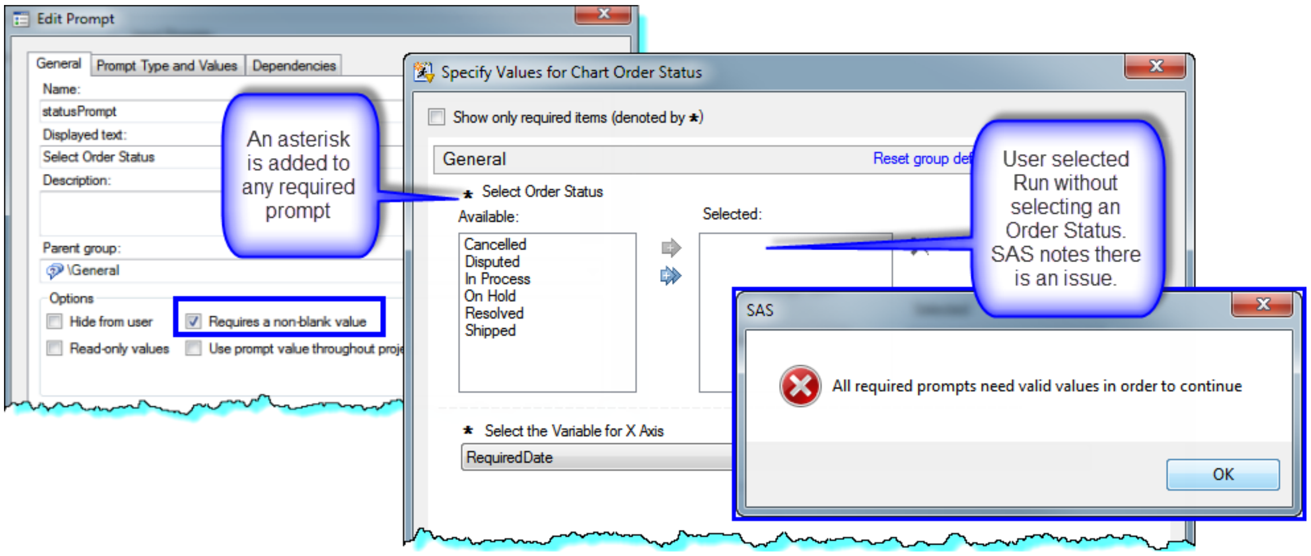
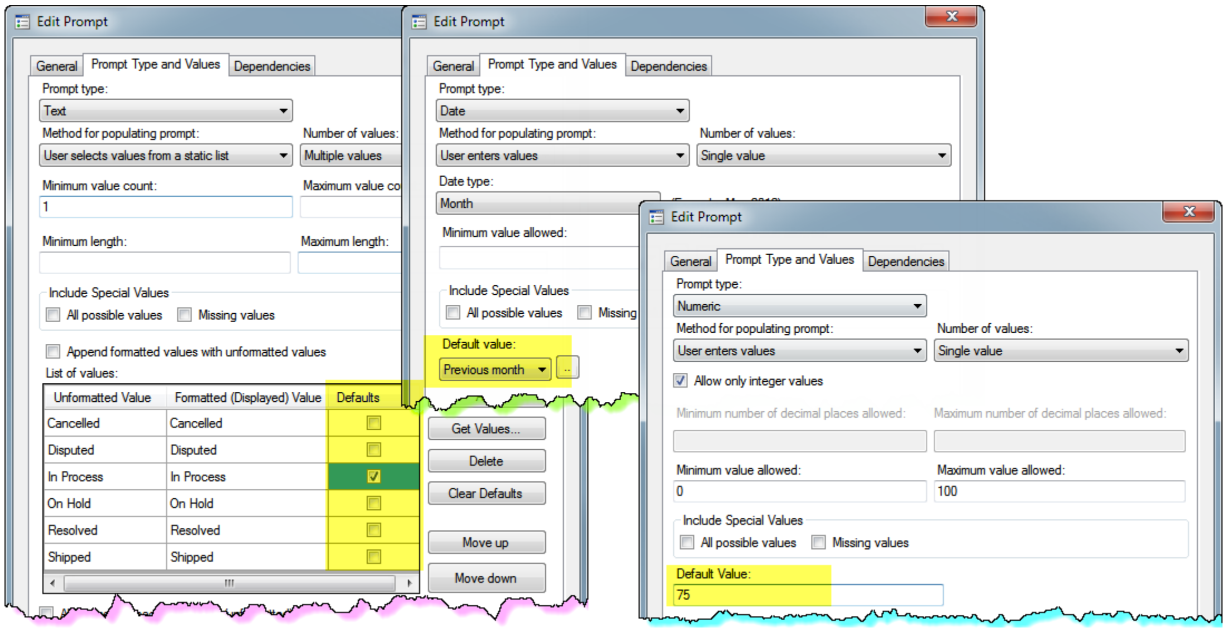

Edit Prompt
General
Prompt Type and Values
Dependencies
Name:
statusPrompt
Displayed text:
Select Order Status
Description:
Parent group:
\General
Options
Hide from user
Read-only values
Requires a non-blank value
Use prompt value throughout proj
An asterisk is added to any required prompt
Specify Values for Chart Order Status
Show only required items (denoted by *)
General
Reset group def
* Select Order Status
Available:
Selected:
Cancelled
Disputed
In Process
On Hold
Resolved
Shipped
* Select the Variable for X Axis
RequiredDate
User selected Run without selecting an Order Status. SAS notes there is an issue.
SAS
All required prompts need valid values in order to continue
OK

Figure 1 Require a non-blank value

DECODER TRICK #2: PROVIDE A DEFAULT VALUE

Tip #1 can be annoying and you might want to make sure the prompt situation warrants it. As an alternative, you can also set a default value for the prompt. Based on the prompt type, there are many ways to indicate the value. This does allow a user to go forward with the defaults so the stored process does not generate an error message. The only downside I see is if the user doesn't understand they can make changes – but most people are computer-savvy enough that they can understand it.

Hint: Set the default value when testing your stored process so you don't have to fill out the values each time.



Edit Prompt

General | Prompt Type and Values | Dependencies

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 co

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
Cancelled	Cancelled	☐
Disputed	Disputed	☐
In Process	In Process	☑
On Hold	On Hold	☐
Resolved	Resolved	☐
Shipped	Shipped	☐

Get Values... | Delete | Clear Defaults | Move up | Move down

Edit Prompt

General | Prompt Type and Values | Dependencies

Prompt type: Date

Method for populating prompt: User enters values

Number of values: Single value

Date type: Month

Minimum value allowed:

Include Special Values: ☐ All possible values ☐ Missing

Default value: Previous month

Edit Prompt

General | Prompt Type and Values | Dependencies

Prompt type: Numeric

Method for populating prompt: User enters values

Number of values: Single value

☑ Allow only integer values

Minimum number of decimal places allowed:

Maximum number of decimal places allowed:

Minimum value allowed: 0

Maximum value allowed: 100

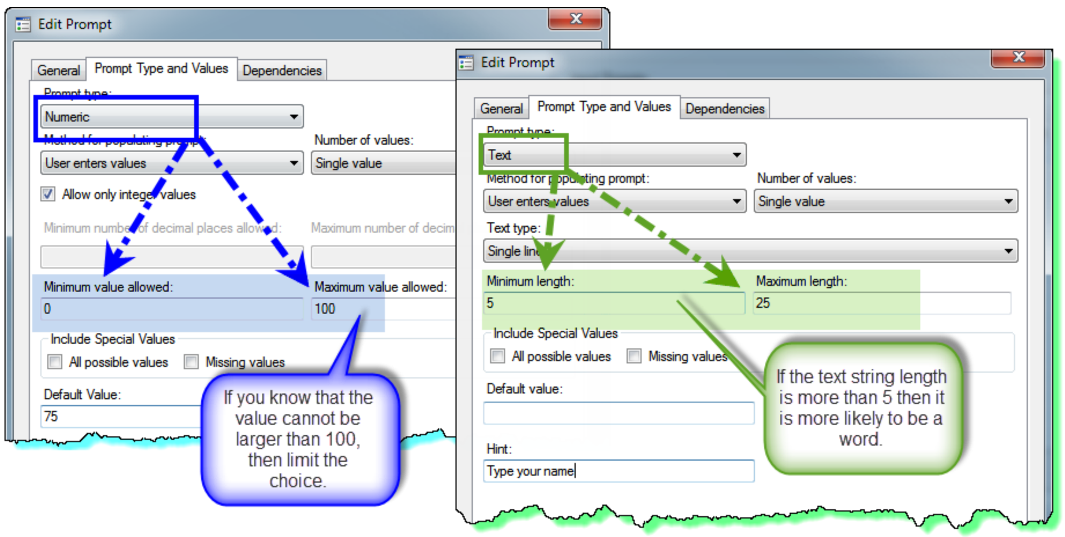
Include Special Values: ☐ All possible values ☐ Missing values

Default Value: 75

Figure 2 Provide default values

DECODER TRICK #3: PROVIDE MINIMUM AND MAXIMUM PROMPT VALUES

If you know, the data starts and ends in a certain period, then don't allow the user to select past those values. Here's some examples to give you some other ideas. For numeric prompts, if you know the user cannot select a value greater than 100, and then limit the value to 100. Likewise, if you are allowing the user to type a value, then set the minimum value to a length of three or five because it is more likely to be a word. You have to decide based on your data – but these examples might generate some thoughts.


Edit Prompt
General
Prompt Type and Values
Dependencies
If you know that the value cannot be larger than 100, then limit the choice.
If the text string length is more than 5 then it is more likely to be a word.

Prompt type: Numeric
Method for populating prompt: User enters values
Number of values: Single value
[✓] Allow only integer values
Minimum number of decimal places allowed:
Maximum number of decim
Minimum value allowed: 0
Maximum value allowed: 100
Include Special Values
[ ] All possible values [ ] Missing values
Default Value: 75

Prompt type: Text
Method for populating prompt: User enters values
Number of values: Single value
Text type: Single line
Minimum length: 5
Maximum length: 25
Include Special Values
[ ] All possible values [ ] Missing values
Default value:
Hint: Type your name

Figure 3 Set minimum and maximum values

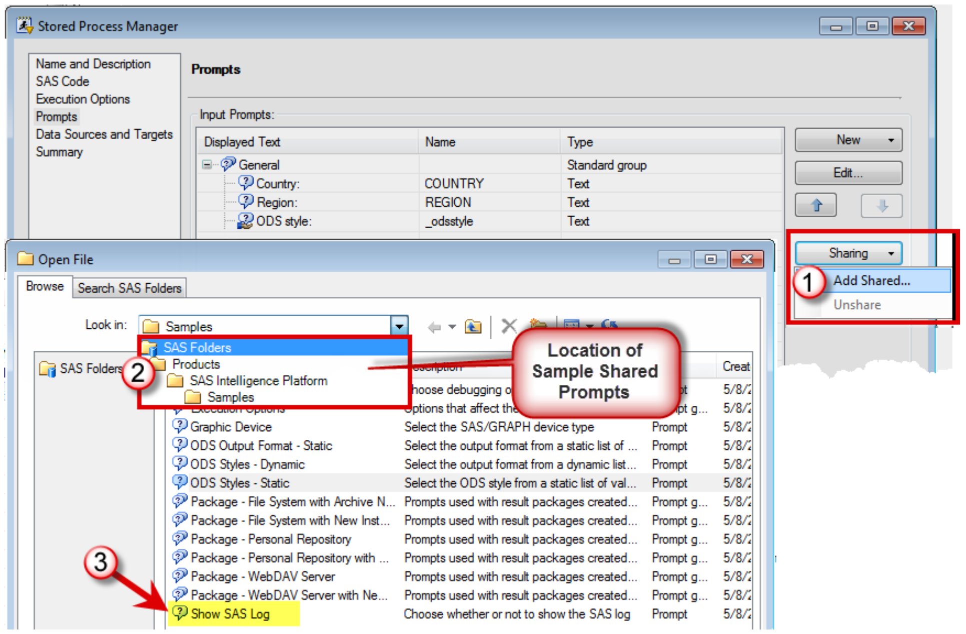
FINDING THE STORED PROCESS LOG

The first statement you will hear from an experienced programmer when you report is an issue is "What did the log show?" Some may not realize that there are several ways to see the log file besides asking the SAS Administrator to retrieve the results.

DECODER TRICK #4: ADD A SUPPLIED SHARED PROMPT

Shared prompts are prompts that someone has already created that can be used with multiple stored processes. SAS ships about 15 shared prompts that allow you to select styles, logs, packages, and so on. Many of the tasks you may routinely need when creating a stored process! These prompts are typically found in the **SAS Folders > Products > SAS Intelligence Platform > Samples** location.

You can add a shared prompt as you create your stored process. From the Prompts pane, select Sharing > Add Shared. Navigate to the shared prompt location and select the Show SAS Log shared prompt.



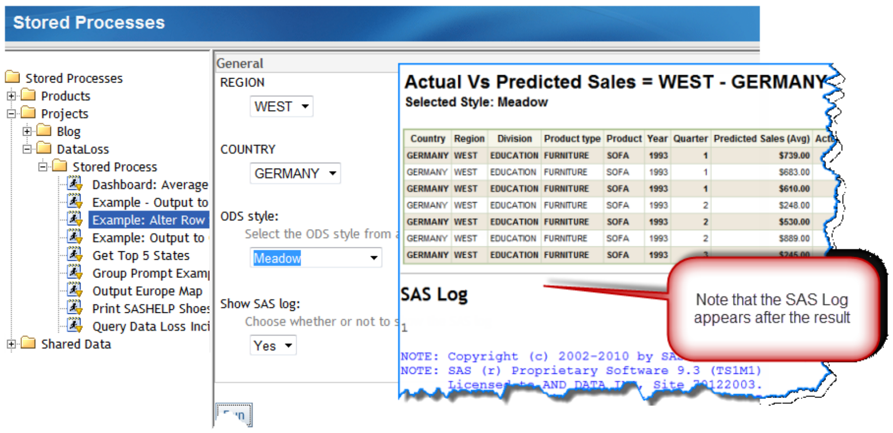
Input Prompts:

Displayed Text	Name	Type
General		Standard group
Country:	COUNTRY	Text
Region:	REGION	Text
ODS style:	_odsstyle	Text

	Description		Creat
	hoose debugging o	pt	5/8/2
Execution Options	ptions that affect the	pt g...	5/8/2
Graphic Device	Select the SAS/GRAPH device type	Prompt	5/8/2
ODS Output Format - Static	Select the output format from a static list of ...	Prompt	5/8/2
ODS Styles - Dynamic	Select the output format from a dynamic list...	Prompt	5/8/2
ODS Styles - Static	Select the ODS style from a static list of val...	Prompt	5/8/2
Package - File System with Archive N...	Prompts used with result packages created...	Prompt g...	5/8/2
Package - File System with New Inst...	Prompts used with result packages created...	Prompt g...	5/8/2
Package - Personal Repository	Prompts used with result packages created...	Prompt g...	5/8/2
Package - Personal Repository with ...	Prompts used with result packages created...	Prompt g...	5/8/2
Package - WebDAV Server	Prompts used with result packages created...	Prompt g...	5/8/2
Package - WebDAV Server with Ne...	Prompts used with result packages created...	Prompt g...	5/8/2
Show SAS Log	Choose whether or not to show the SAS log	Prompt	5/8/2

Figure 4 Adding a Shared Prompt

If you answer the prompt as Yes, then the log displays after the results.



REGION: WEST

COUNTRY: GERMANY

ODS style: Meadow

Show SAS log: Yes

Actual Vs Predicted Sales = WEST - GERMANY

Selected Style: Meadow

Country	Region	Division	Product type	Product	Year	Quarter	Predicted Sales (Avg)	Act
GERMANY	WEST	EDUCATION	FURNITURE	SOFA	1993	1	$739.00	
GERMANY	WEST	EDUCATION	FURNITURE	SOFA	1993	1	$683.00	
GERMANY	WEST	EDUCATION	FURNITURE	SOFA	1993	1	$610.00	
GERMANY	WEST	EDUCATION	FURNITURE	SOFA	1993	2	$248.00	
GERMANY	WEST	EDUCATION	FURNITURE	SOFA	1993	2	$530.00	
GERMANY	WEST	EDUCATION	FURNITURE	SOFA	1993	2	$889.00	
GERMANY	WEST	EDUCATION	FURNITURE	SOFA	1993	3	$245.00	

SAS Log

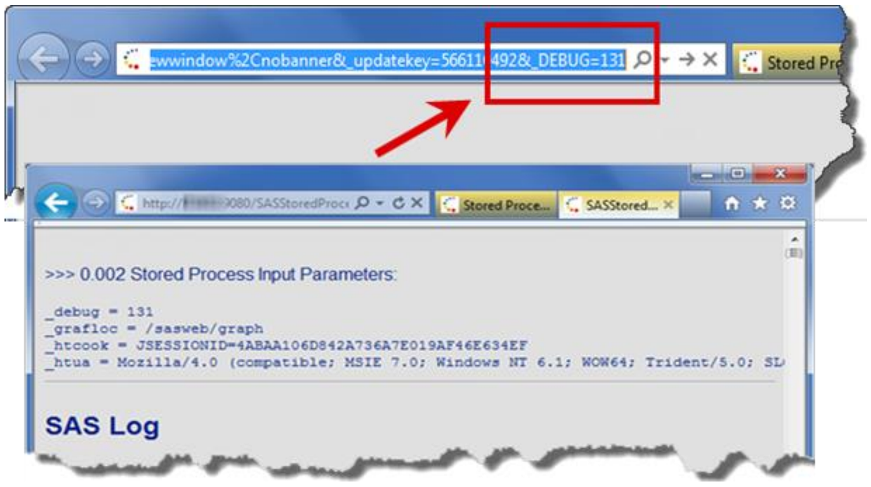
```
1

NOTE: Copyright (c) 2002-2010 by SAS
NOTE: SAS (r) Proprietary Software 9.3 (TS1M1)
      Licensed to AND DATA INC, Site 70122003.
```

Figure 5 Log appears after the results

DECODER TRICK #5: SKILLFULLY APPLY THE URL

When the stored process fails, a standard error message appears with a button to open the SAS log. If stored process results are not returned as expected, the log may not appear either. You do not know what caused your stored process to visit La-La land. Rather than asking the SAS administrator to get the most recent log off the server, you can add a parameter to the URL path to see the log quickly.

From the SAS Stored Process Web Application, add the **&_DEBUG=131** option to the end of the stored process path and press Enter, as shown in the following figure. After the stored process runs, the log appears in the Web browser as shown below. Now you can locate the errors in your mis-behaving stored process.


ewwindow%2Cnobanner&_updatekey=566110492&_DEBUG=131
Stored Pro
http://[redacted]9080/SASStoredProc
Stored Proce...
SASStored...

>>> 0.002 Stored Process Input Parameters:

```
_debug = 131
_grafloc = /sasweb/graph
_htcook = JSESSIONID=4ABAA106D842A736A7E019AF46E634EF
_htua = Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/5.0; SL
```

SAS Log

Figure 6 Add the Debug Option

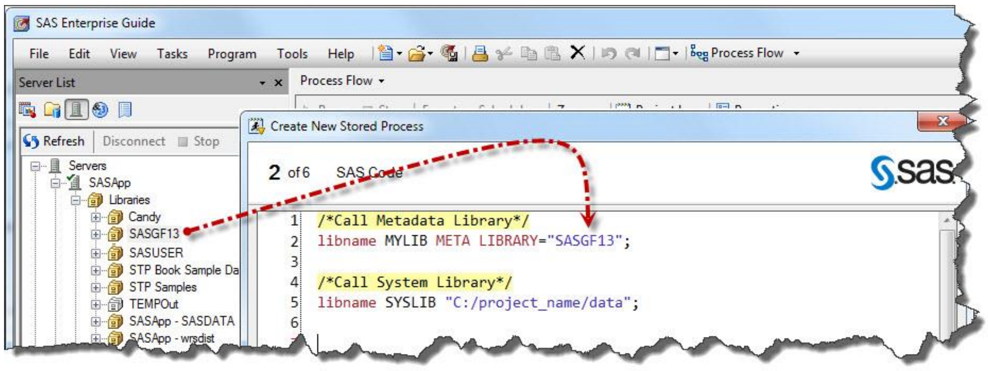
REVIEWING COMMON STORED PROCESS ERRORS

Here are the most common errors that new developers make to help you as you get started.

DECODER TRICK #6: AVOIDING LIBNAME NOT FOUND ERRORS

If you develop your stored process code in SAS Enterprise Guide, then the code uses your connection profile. Thus, it has access to all of your libraries. When the same code runs from the stored process, you might find it generates a “LIBNAME not found error”. We recommend adding a META libname statement to your stored process, however it is best to contact your SAS Administrator to determine which approach will work in your SAS environment.

You can use a metadata library or use a system library. If you use a system library, it needs to be one that the target user group can access. For instance, you wouldn’t want to use “C:/My Secret Folders” as a path since only you can access the location.



```
/*Call Metadata Library*/
libname MYLIB META LIBRARY="SASGF13";

/*Call System Library*/
libname SYSLIB "C:/project_name/data";
```

Figure 7 Add a LIBNAME statement

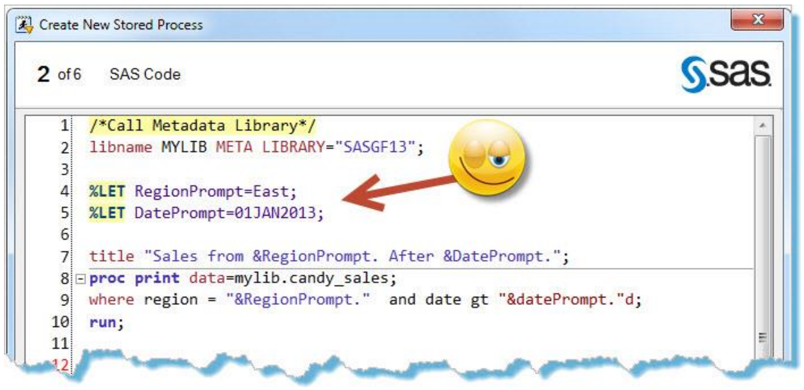
DECODER TRICK #7: AVOIDING UNEXPECTED RESULTS

If you have added the %STPBegin and %STPEnd macros in your stored process, then you must ensure the Stored Process Wizard does not add it back. If you have added the macros and the wizard has added the macros, then you see where some interesting results might occur. The worse part of troubleshooting the issue is that it can take several hours with the log to determine the issue. The best approach is to not only review the logs, but double check the .sas file that exists on the server. The authors also recommend using EG to create the Stored Process. But only editing the code directly on the .sas file that exists on the server.

DECODER TRICK #8: AVOID THE PROMPT ALWAYS USING THE SAME VALUES

Did your prompt work but it keeps making the same choice no matter what you select? For instance, you selected the East region but it returns the West region – no matter how many times you select it? You might want to double-check the code to see if you left the LET statements in it. Many programmers test the prompts to ensure they work or create errors to confirm the prompt will fail.

Tip: Use %Let statements to test your prompts – but be sure to remove them!



```
/*Call Metadata Library*/
libname MYLIB META LIBRARY="SASGF13";

%LET RegionPrompt=East;
%LET DatePrompt=01JAN2013;

title "Sales from &RegionPrompt. After &DatePrompt.";
proc print data=mylib.candy_sales;
where region = "&RegionPrompt."  and date gt "&datePrompt."d;
run;
```

Figure 8 Look for test code left behind

DECODER TRICK #9: REDIRECT THE LOG FILE TO ANOTHER LOCATION ON THE SERVER

There are occasions where the stored process log is in a location that is inaccessible by developers. Or when the log is so large from so many developers that finding your errors could take some time. Use the PROC PRINTTO code to redirect the output to another location.

Place above the %stpbegin:

```
proc printto print = 'c:\My Secret Location\logmessages.txt' new; run;
```

Place at the end of the program:

```
proc printto; run;
```

Tip: Just don't forget to comment or remove this code as it could impact other users if they don't have access to the same secret location.

DECODER TRICK #10: MAKE USE OF MACRO PRINT OPTIONS

When using macros within the code, adding options such as MPRINT and MLOGIC will increase the amount of logging show. This can come into handy when trying to determine what is testing as true or false in %if statements.

```
options mprint mlogic;
```

Without Use	With Use

`10   %global test;`
`11   %macro t;`
`12   %if &test = True %then %do;`
`13   %put test passed;`
`14   %end;`
`15   %else %do;`
`16   %put test failed;`
`17   %end;`
`18   %mend;`
`19   %t;`
`test failed`
	`20   options mprint mlogic;`
	`21   %global test;`
	`22   %macro t;`
	`23   %if &test = True %then %do;`
	`24   %put test passed;`
	`25   %end;`
	`26   %else %do;`
	`27   %put test failed;`
	`28   %end;`
	`29   %mend;`
	`30   %t;`
	`MLOGIC(T):  Beginning execution.`
	`MLOGIC(T):  %IF condition &test = True is FALSE`
	`MLOGIC(T):  %PUT test failed`
	`test failed`
	`MLOGIC(T):  Ending execution.`

TUNING A STORED PROCESS

The same tricks you use to tune other SAS code applies equally to your stored process code. Here's a few examples:

- Test if the prompts have needed values and if they do not, then end the process early.
- Avoid making multiple trips to the database if you can extract the needed data all at once.
- Try to minimize the data by using Keep and Drop statements.
- If it's a popular stored process, it might make more sense to extract the data during the a batch process. This way the data could be prepared earlier in the day and be ready for queries.
- Index the key variables in larger datasets to improve the retrieval speed.

REFERENCES

Aanderud, Tricia and Hall, Angela. 2013. Creating Your First SAS Stored Process. SAS Global Forum 2013 Proceedings. Available at http://support.sas.com/resources/papers/proceedings13/148-2013.pdf

Aanderud, Tricia and Hall, Angela. 2012. The 50 Keys to Learning SAS Stored Processes. Raleigh, NC: Siamese Publishing.

ACKNOWLEDGMENTS

The authors would like to thank Vince DelGobbo of SAS for reviewing our paper and providing valuable insight into additional troubleshooting techniques.

RECOMMENDED READING

Aanderud, Tricia and Hall, Angela. 2012. Building Business Intelligence with SAS. Cary, NC: SAS Press.

BI Notes for SAS Users, Aanderud, Tricia. Available at http://www.bi-notes.com

Real BI for SAS Users, Hall, Angela. Available at http://blogs.sas.com/content/bi/

Lafler, Kirk Paul, 2012. Top Ten SAS® Performance Tuning Techniques. Available at:
http://support.sas.com/resources/papers/proceedings12/357-2012.pdf

CONTACT INFORMATION

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

Tricia Aanderud
And Data Inc
tricia.aanderud@and-data.com

Angela Hall
SAS Institute
angela.hall@sas.com