Paper RIV-07

Case Study: Migrating an Existing SAS® Process to Run on the SAS Intelligence Platform

Joseph Urbi, WellPoint, Virginia Beach, VA

ABSTRACT

In 2012, the WellPoint Medicaid Business Unit (formerly Amerigroup Corporation) installed components of the SAS Intelligence Platform and moved away from what was primarily a desktop SAS environment. We sought to take advantage of the enhanced capabilities of the SAS Intelligence Platform and also enable new technologies such as SAS Enterprise Miner ™. However, this meant migrating a significant SAS application, the Chronic Illness Intensity Index (CI^3), to run in SAS Enterprise Guide®. The CI^3 Reports Package present the results of the Wellpoint Medicaid Business Unit Continuous Case Finding prioritization process and improve efficiency by providing relevant clinical information about members to drive medical management activities. This paper describes the lessons learned and points out some best practices in updating a legacy SAS application to run in a new environment.

INTRODUCTION

BACKGROUND

The Chronic Illness Intensity Index (CI^3) Process is the umbrella name given to the workflow used to create the CI^3 Database and CI^3 Report Package, which are the products of our monthly Continuous Case Finding (CCF) prioritization process. These products improve efficiency by providing relevant clinical information about the members to drive medical management activities. The CI^3 list displays member-level clinical indicators including overall CI^3 score, other risk based model scores, and plan rank for each member in the plan and includes supporting demographics, diagnostics, utilization, and case management data. The CI^3 Report Package is a set of Excel workbooks which show case management enrollment statistics, as well as aggregated outcomes from the CI^3 database. The CI^3 output is used throughout the organization.

The CI^3 process was originally developed and run in SAS Display Manager and had grown organically over the years as additional data and indicators were added to the original content. Many developers and business analysts had enhanced the SAS programs during that time, with varying coding styles and documentation standards. It was becoming increasingly difficult to modify the SAS code because it had become so fragmented in style. In addition, the overall CI^3 process was taking over 130 hours to complete, which made it difficult to meet deadlines when upstream processes were delayed. In 2012, WellPoint Medicaid Business Unit management decided to move to a server based SAS platform, with SAS Enterprise Guide as the main client software. It also decided that the time had come to redo the core CI^3 process to make it more maintainable and shorten the time to run the process.

This paper focuses on the evolution of the CI^3 Core Process code, which is a key task within the overall CI^3 process. Similar changes have been applied to other SAS programs involved in the process with the same positive results.

APPROACH

The temptation when migrating to a new software environment, especially one as feature rich as SAS Enterprise Guide, is to start over from scratch, so that you can take advantage of ALL the new goodies that are now at your disposal. Often this effort takes months, and the only time you have a working system is when all the modules are completed and delivered. This form of development carries a lot of risk with regard to delivering required functionality and meeting schedule.

Another possible approach is to use a feature of SAS Enterprise Guide called *Analyze Program Flow* that reads your SAS code and creates a new process flow as the result. Our testing revealed that this feature worked best when the program analyzed was fairly small and well coded. Running it against the 3000 lines of code that comprised the CI^3 Core Process resulted in an indecipherable web of tasks that would have taken a long time to straighten out.

Because the CI^3 output was produced monthly and also because of a looming license expiration for SAS Display Manager, the development team chose a different approach. We would incrementally migrate the CI^3 code base to SAS Enterprise Guide, making sure that we always had a functioning system that could deliver the required output every month. We also decided to initially take advantage of only a few of the new capabilities offered by SAS Enterprise Guide – those that offered us the most value.

FIRST STEPS

Our first step was to take the CI^3 Core Process code and to run it under SAS Enterprise Guide unchanged. We did this because we had a whole new SAS infrastructure and we wanted to identify as early as possible the problems that the new environment would impose on us.

Right away, we identified two issues that kept us from running the CI^3 code in SAS Enterprise Guide right out of the box. The first issue related to drive mappings and the second related to the different default handling of variable names between SAS Display Manager and SAS Enterprise Guide.

The HCE department uses a drive mapping to point to folders on the network which hold shared data, in this case the "O:" drive. Other departments' workstations within the WellPoint Medicaid Business Unit also map to a common "O:" drive, but very often they point to a different physical storage location. Because of this, a common drive mapping on the SAS Servers is not feasible and UNC naming conventions have to be used. Thus, all the CI^3 code that referred to the "O:" drive had to be changed to the form "\\Server\Share\.... "

The second problem was harder to identify. There are several PROC TRANSPOSE procedures in the CI^3 code. A couple of these transpose procedures relied on a character field for the names as the resulting columns. The problem occurred when the field that was to become the column name had an embedded space in it. The default behavior in SAS Display Manager 9.3 is to replace the space with an underscore, resulting in a valid SAS variable name. However, in SAS Enterprise Guide, the default behavior is to allow embedded spaces and special characters in variable names. This caused problems in subsequent steps of the process that relied on the underscore character being part of the variable name. The solution was to set the system level option to VALIDVARNAME=V7, which forced SAS Enterprise Guide to handle variable names the same as SAS Display Manager 9.3.

Once these two issues were resolved, the program could be run successfully in SAS Enterprise Guide.

IMPROVE THE CODE

We could have stopped after we got the CI^3 code running in SAS Enterprise Guide and creating correct output; however, just because you are porting a program to Enterprise Guide doesn't mean that you should bring along all the deficient code that has accumulated over time. For example, consider the following code that was found in the original CI^3 program and then the reworked code below it.

```
/* Original code fragment from CI3 */
data LTSSa;
set LTSS;
keep Member system userid begin_dte;
run;
data LTSSa;
set LTSSa;
format begin_dt mmddyy10.;
begin_dt=datepart(begin_dte);
run;

/* Reworked code fragment using a single data step */
data LTSSa;
       format begin_dt mmddyy10.;
       set LTSS (keep=Member system userid begin_dte);
       begin_dt=datepart(begin_dte);
run;
```

Having both data steps was not necessary and could be combined. Not only is the reworked code more legible, but also is more efficient because we eliminate an additional pass through the same data. While this may not matter in smaller SAS programs, it can add a lot of processing time when the data sets are large and there are many redundant data steps.

Another way to improve SAS code is to use more advanced data structures like arrays and hash tables. For example, the original CI^3 SAS code had a macro named zeros defined as below.

```
%macro zeros(util);
data ccf5_&time1.; set ccf5_&time1.;
if &util._1=. then &util._1=0;
if &util._2=. then &util._2=0;
if &util._3=. then &util._3=0;
if &util._4=. then &util._4=0;
if &util._5=. then &util._5=0;
```

```
if &util._6=. then &util._6=0;
if &util._7=. then &util._7=0;
if &util._8=. then &util._8=0;
if &util._9=. then &util._9=0;
if &util._10=. then &util._10=0;
if &util._11=. then &util._11=0;
if &util._12=. then &util._12=0;
run;
%mend zeros;

%zeros(SPEC);%zeros(PCP);%zeros(OPER);%zeros(DAYS);%zeros(IP);
```

This code is not very flexible. It requires 12 values and only operates on a "single" variable at a time. In fact, similar macros were found elsewhere in the code because the variables they were supposed to "zero out" had a different number of values or did not meet the naming convention defined in the macro. We replaced all of them with the following macro, which can be found in Gerhard Svolba's excellent book, "Data Preparation for Analytics".

```
%macro replace_mv(cols,mv=.,rplc=0);
       array varlist {*}  &cols;
       do _i = 1 to dim(varlist);
         if varlist{_i} = &mv then varlist{_i}=&rplc;
       end;
       drop _i;
%mend;

data ccf5_&time1.;
       set ccf5_&time1.;
       %replace_mv(SPEC_: PCP_: OPER_: DAYS_: IP_:);
run;
```

This macro demonstrates the use of an array that self-dimensions based on the number of columns presented to it. We are no longer locked into a specific number of occurrences of the variable and we can also process multiple variables at a time, eliminating extra passes through the same data. Note that in this code we used name prefix lists; we could also have used numbered range lists as well (e.g., SPEC_1 – SPEC_12).

There are many other techniques to improve SAS code. As Art Carpenter notes, through SAS "we have the ability to solve most problems in more than one way, and the various solutions will not all have the same efficiency (for the computer or for the programmer)." The key is to continually learn new techniques from other SAS programmers and through resources such as books and published papers.

SIMPLIFY THE PROCESS

Once the SAS code base had been made more maintainable, it was time to turn our attention toward improving the overall process flow. Here, we looked at things such as eliminating manual entry, breaking up or rearranging the order of processing, adding the capability to restart the process in the middle, etc., that would result in the overall process taking less time and/or run more efficiently. This is where we looked to take advantage of two key features of SAS Enterprise Guide: Prompting and Process Flows.

PROMPTING

Before the CI^3 SAS programs could be run, the business analyst had to edit the code to change a number of macro variables, many of which involved dates. The snippet of code to be edited looked like this.

```
*** This is the date that will be the name of the CI3 EXCEL lists;
%let listmnth=October2012;
*** This is the date run month for the RO file;
%let MONTH_RUN=OCT2012;
*** DMCCU Report Month;
%let DMCCUMOR="10/01/2012";

*** and so on ;
```

In all, there were seventeen macro variable assignments related to dates. Upon further analysis, all the dates really are constant offsets from or varying formats of a single date – CI^3 Report Month. Once we realized this, we decided that we could use the Enterprise Guide prompting facility to obtain the CI^3 Report Month date and then calculate the

remaining date variables in a macro. The first step toward doing this was writing the macro code that would calculate all the other dates.

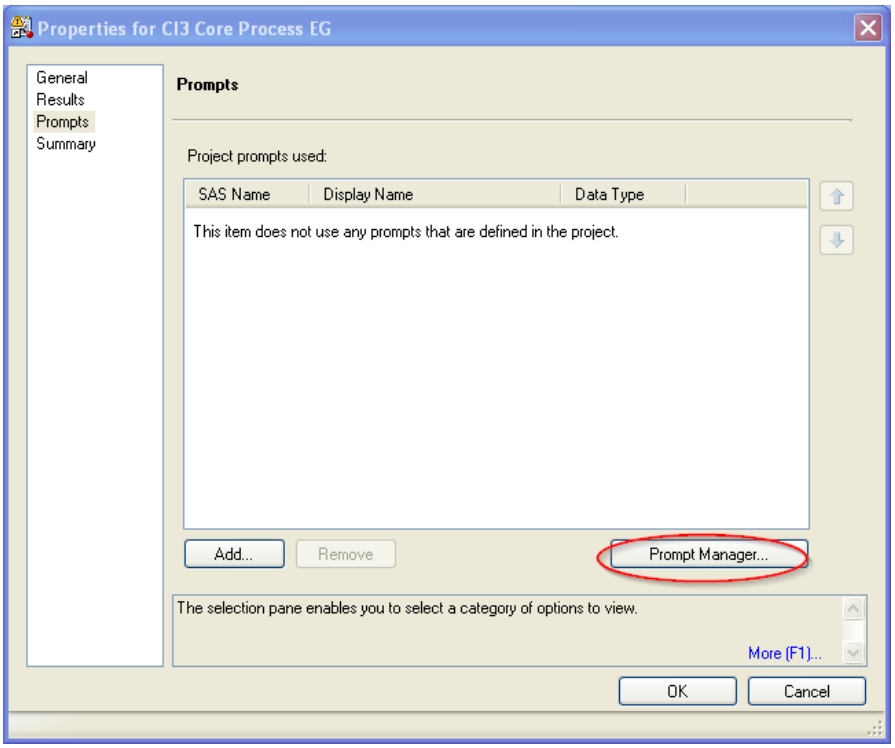
```
/* Create macro variables that will become EG prompts. Later we'll comment out
   after prompts are set up
*/
%let ci3_month = October;
%let ci3_year = 2012;

/* Calculate a SAS date from ci3_month and ci3_year to use for other dates */
%let rpt_month = %sysfunc(inputn(%sysfunc(cats(01, &ci3_month, &ci3_year)), DATE9.));

/* We don't create an output data set, so use data _NULL_ to set other macro variables
*/
data _NULL_;
  call symput('listmnth', cats(put(&rpt_month., WORDDATX9.),put(&rpt_month., YEAR.)));
  call symput('MONTH_RUN', put(&rpt_month., MONYY7.));
  call symput('DMCCUMOR', put(&rpt_month., MMDDYY10.));
run;
```

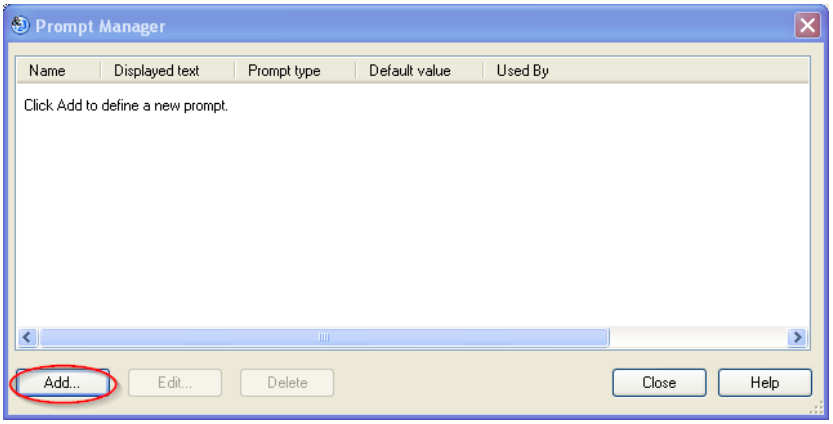
Once this was done, prompts could be created which would obtain the CI^3 Report Month date from the user. SAS Enterprise Guide has a number of prompt types available, including a Date prompt. Because the CI^3 Report Month date is always the first of the month, we decided not to use the built in Date prompt since it would allow the user to select other days of the month. Thus we created separate month and year variables that were used to create a valid CI^3 Report Month date.

To create prompts, you need to invoke the SAS Enterprise Guide Prompt Manager. The easiest way to do this is through the Program Properties dialog box as seen in Display 1. To view this dialog, right click on your program task in the process flow and select Properties. Highlight the Prompts link in the top left corner of the dialog, then press the Prompt Manager button.



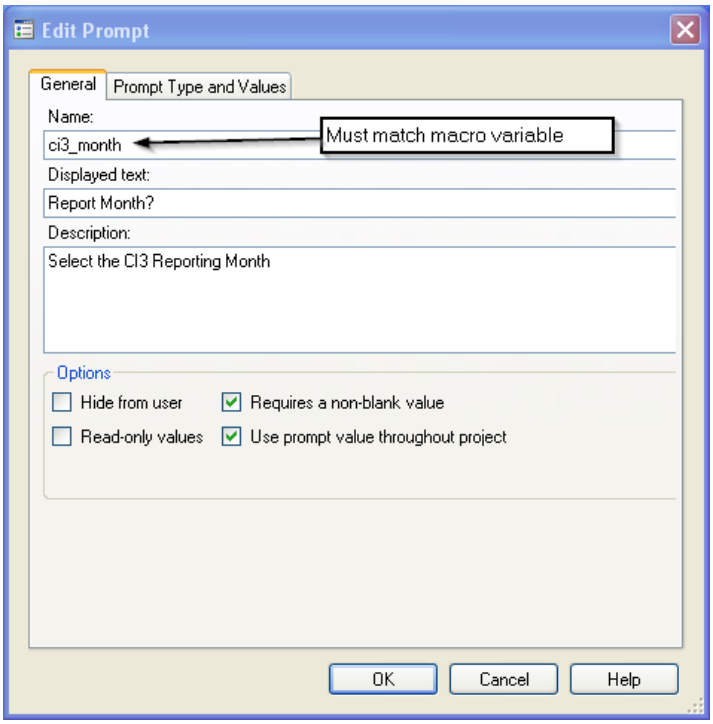
Display 1. Program Properties Dialog Box

You will be presented with the Prompt Manger dialog box as shown in Display 2. This dialog box lists the prompts that have already been defined in your SAS Enterprise Guide project. Press the Add button, which will open the Edit Prompt dialog where you will define a new prompt.



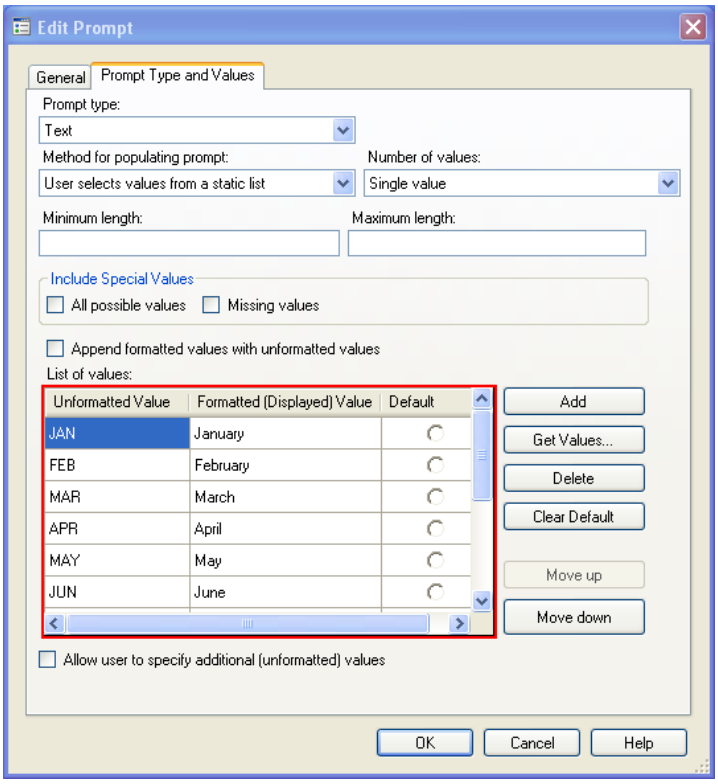
Display 2. Prompt Manager Dialog Box

For example, to define a new prompt for the ci3_month macro variable, you would fill out the Edit Prompt dialog box as shown in Display 3. The Name field is what connects the prompt to the macro variable, so it must match the macro variable that you are prompting for.



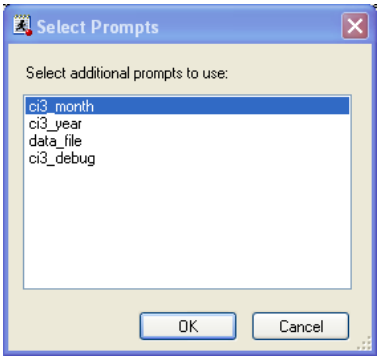
Display 3. Edit Prompt Dialog – General Tab

Next, press the Prompt Type and Values tab. For our application, we want the ci3_month macro variable to contain a text value. Since we don't want the user to type in the month name, under "Method for populating prompt", we choose "User selects values from a static list". Finally, we enter the valid months in the List of Values box. See Display 4. Note that we are displaying the full month name (e.g. "January"), but returning the month acronym ("JAN") to the macro variable.



Display 4. Edit Prompt Dialog – Prompt Type and Values Tab

Press the OK button and you will be returned to Properties Dialog (See Display 1). Press the Add button and you will be presented with the Select Prompts dialog box (Display 5). Press OK and now the prompt and the macro variable are associated with each other.



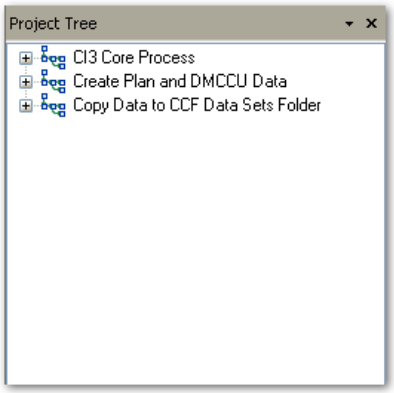
Display 5. Select Prompts Dialog

You would follow the same steps to define a prompt for the ci3_year and any other macro variables that you want to use in your application.

PROCESS FLOWS

One of the advantages of using a tool like SAS Enterprise Guide is the ability to organize a project into several smaller process flows. When starting a new project, it is quite easy to design logical breaks in the processing to take advantage of this capability. However, taking a legacy SAS program like the CI^3 Core Process posed some challenges. Because it had grown organically over time, related processing steps were spread throughout the code. While we could have left the program in this state, one of our design goals was to break the program up into more logical pieces that would give us the ability to start the CI^3 Process at any step along the way without running prior

steps. After much trial and error, we divided the code into three main processes, which can be seen in the Project Tree screenshot in Display 6.



Display 6. CI^3 Project Tree in SAS Enterprise Guide

Once we did this, we needed a way to save/recall the key macro parameters and also restore the work library to where it was at the end of the previous step so that we could restart any process.

Here is the code we used to save the macro variables.

```
/* save the macro variables for use in later programs */
data stg.ci3vars ;
       set sashelp.vmacro(where=(scope='GLOBAL'));
       if upcase(name) in ('CI3_MONTH','CI3_YEAR','CCFLIB','TIME1','TIME2',
                           'TIME3','TIME4', 'DATA_FILE','LISTMNTH','CT_DT',
                           'TOOLBOXLIB','DMCCUMOR','DATA_MONTH','RPT_MONTH',
                           'PCP_MONTH','MONTH_RUN','DS_SUFFIX', 'CI3_DEBUG', 'DEST');
run;
```

The sashelp.vmacro view contains information about currently defined macros. This data step queries the view for GLOBAL macro variables, then outputs only those that apply to our application to the ci3vars data set in permanent library named stg.

Now that we have the macro variables stored, we need some code to retrieve the macro variables and restore the WORK library to a known state. Note that we check to see if the data set already exists in the WORK library to avoid unnecessarily moving the data across the network. This code is placed in a program task located in each process flow that might need to be restarted.

```
/* Get the macro variables from the CI3_Core_Process run */
data _null_;
       set stg.ci3vars (where=(scope='GLOBAL'));
       if substr(name,1,3) NE 'SYS' then do;
              call execute('%global '||strip(name)||';');
              call execute('%let '||strip(name)||'='||strip(value)||';');
       end;
run;

/*Check for ccf_final data set in work library. Copy if not found */
%macro ccf_final_exists;
       %if %sysfunc(exist(work.ccf_&time1._final)) = 0 %then %do;
              proc datasets nolist memtype=data;
                     copy in=stg out=work;
                            select ccf_&ds_suffix._final;
              run;
       %end;
%mend ccf_final_exists;

%ccf_final_exists
```

SAS Enterprise Guide provides a number of other capabilities that help you improve your overall processing. Get more information from the papers listed in the recommended reading section.

MAKE IT FASTER

After porting the CI^3 Core Process code to SAS Enterprise Guide and improving how the process was run, it was time to address the processing speed. It is always important to tune stable code for performance; otherwise your tuning may become invalid after you make structural changes to your code.

Just moving the application to server based SAS did a lot to improve CI^3 processing time. Since the servers were co-located at the data center with the various data repositories, we had already seen an improvement in processing time because data did not have to travel over a wide area network to a single workstation. Still, we believed that there were better times that could be gained. We used the SAS logs (See Output 1) to benchmark individual tasks and determine where our tuning efforts would be most fruitful. Using the FULLSTIMER option gave us more detailed information in the log that was useful for benchmarking purposes.

```
…
NOTE: There were 2212540 observations read from the data set WORK.CLAIMS.
NOTE: The data set STAGE.CLAIM_SUMMARY has 2212540 observations and 10 variables.
NOTE: Compressing data set STAGE.CLAIM_SUMMARY decreased size by 45.68 percent.
      Compressed is 11899 pages; un-compressed would require 21907 pages.
NOTE: DATA statement used (Total process time):
      real time           21.59 seconds
      user cpu time       2.57 seconds
      system cpu time     0.62 seconds
      memory              434.87k
      OS Memory           14628.00k
      Timestamp           07/31/2012 05:05:58 PM
…
```

Output 1. Partial Log from CI^3 Process with FULLSTIMER option

For our application, we found that most of the longest steps in our process involved reading data from SAS data sets and SQL Server. Once we knew the areas to focus on, it was trial and error to determine the optimum techniques we needed to use to improve these long running tasks. It turned out that setting the READBUFF parameter in the appropriate SET and LIBNAME statements markedly improved the read performance. It is beyond the scope of this paper to go into details about performance tuning, but there are numerous publications to help you. The key is to interpret the log and know where to spend your efforts.

RESULTS

After completing these changes to the CI^3 Core Process code, processing times for this section dropped from an average of 20 hours to 51 minutes. Similar changes to the rest of the CI^3 code base reduced the overall processing time to 56 hours, an overall reduction of 74 hours from the previous average time to process of 130 hours.

Just as important, our incremental project approach resulted in better software. We improved the original code to be more maintainable and stable. By taking advantage of couple of key SAS Enterprise Guide features, we were able to simplify the process which reduced run time errors and introduced the ability to reliably restart the process without having to go back to the beginning. An added bonus of migrating the code to a SAS Enterprise Guide project is that the process flows are self-documenting, making change easier in the future.

An effort like this is never complete and we look forward to taking advantage of more SAS Enterprise Guide features, such as prebuilt tasks and stored processes.

ACKNOWLEDGMENTS

Thanks to Michael Ross for his time in reviewing this paper and making it a better product.

REFERENCES

Svolba, Gerhard. November 2006. *Data Preparation for Analytics Using SAS*. Pg. 127 - 128. Cary, NC: SAS Institute Inc.

Carpenter, Art. 2013. "'How Do I …?' There is more than one way to solve that problem; Why continuing to learn is so important." *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc.

RECOMMENDED READING

Fecht, Marje and Dhillon, Rupinder. 2012. SAS Enterprise Guide 4.3: Finally a Programmer's Tool." *Proceedings of the SAS Global Forum 2012 Conference*. Cary, NC: SAS Institute Inc.

Hall, Angela. 2011. "Creating Reusable Programs by Using SAS Enterprise Guide Prompt Manager." *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc.

Polak, Leonard. 2012. "It's Now Your Project—Clean It Up and Make It Shine." *Proceedings of the SAS Global Forum 2012 Conference*. Cary, NC: SAS Institute Inc.

Ravenna, Andy. 2011. "Becoming a Better Programmer with SAS Enterprise Guide 4.3." *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Joe Urbi
WellPoint, Inc.
4425 Corporation Lane
Virginia Beach, VA 23462
joseph.urbi@wellpoint.com