Paper AD-35

Programming Compliance Made Easy with a Time Saving Toolbox

Patricia Guldin, Merck & Co., Inc., Kenilworth, NJ USA

ABSTRACT

Programmers perform validation in accordance with established regulations, guidelines, policies and procedures to ensure the integrity of analyses and reporting, reduce risk for delays in product approvals, fines, legal actions, and to safeguard reputations. We understand the importance, but the time involved to produce and appropriately store the documentation and evidence required to prove we followed process and SOPs can be labor intensive and burdensome. Using SAS/AF®, SAS® Component Language and .NET we have developed two versions of an automated tool that can be used with PC SAS® or Enterprise Guide®. The toolbox is designed to make compliance with programming SOPs easier, increase consistency, and save the programmer time. The toolbox auto-populates some information and saves documentation in designated locations as actions are performed. Functions include creating and verifying a standard program header, updating program headers, revision history and version date, creating validation environments including testing checklists, and promoting programs. The toolbox is also used to view transaction logs, create and/or generate batch jobs for remote execution in UNIX, and to select and include macro calls from a macro library.

INTRODUCTION

In 2002, Merck created a programming toolbar to help support compliance. It was compatible with PC SAS and was designed around the processes in place at that time. Over time, the toolbar was no longer compatible with a new programming environment, processes and technology and use of the toolbar declined. Merck recently developed a programming compliance toolbox which is a user-friendly interactive tool used in a SAS editor to assist the programmer with documentation and compliance to SOPs and processes. It is the result of global collaboration between Merck programmers in Belgium, China, Japan, and the US. The toolbox is designed to function within the context of Merck statistical programming SOPs and processes, and to work in connection with a UNIX based reporting platform with a standard directory structure. It can be used with PC SAS or Enterprise Guide. Many of the functions of the old toolbar were incorporated and additional functionalities of promotion to production and batch jobs were added. The toolbox automatically saves items, such as programs and validation documentation, in folders defined by the standard directory structure. Functions of the toolbox include creating and updating program headers, creating validation environments and testing checklists and promoting programs. The toolbox is able to auto-populate certain items by saving information such as a programmer name, programmer id, and SAS version in a sasuser.profile. The toolbox code pulls the required information as needed from the profile for each of the toolbox functions. The toolbox is also used to view transaction logs, create and/or generate batch jobs, insert proc dataset delete, pick up macros, insert program steps and update program flow for those steps.

INSTALLING THE TOOLBOX

SAS ENTERPRISE GUIDE

The Enterprise Guide version of the toolbox uses Microsoft .Net. The core component is a DLL file that is placed in a specific folder and is integrated with SAS. Oracle 11g 64 bit is required. When a programmer is in Enterprise Guide, they select Tools, Add-In, and CPI Programming Compliance Toolbox to open and start using the toolbox.

PC SAS

Oracle 11g 32 bit is required for the PC SAS version of the toolbox. A programmer executes a %toolbox macro to install the toolbox .This macro copies the toolbox catalog files to the SASUSER library. The catalog files contain the frames and code for each function, the user profile, and the settings for the toolbox icons. A default toolbar containing the icons for the toolbox functions is installed in the enhanced editor for the programmer.

USING THE TOOLBOX

Each function of the toolbox, except Proc Datasets Delete and Insert Step, has a user-friendly screen or set of screens which the programmer uses to perform that function. The programmer selects the icon for the function desired and a screen or form opens. The programmer works through the screen, entering, selecting, or verifying information needed to complete the function.

A macro variable (PROTPATH) must be defined on UNIX for proper functioning of the toolbox. This variable defines the path to the high level folder for a particular deliverable or reporting event. Without defining this variable the toolbox will not know where to start and where to save items. Additionally, available macro library locations must be defined using SASAUTOS. Applicable information for each function is prepopulated for the programmer using the value of protpath and the user profile.

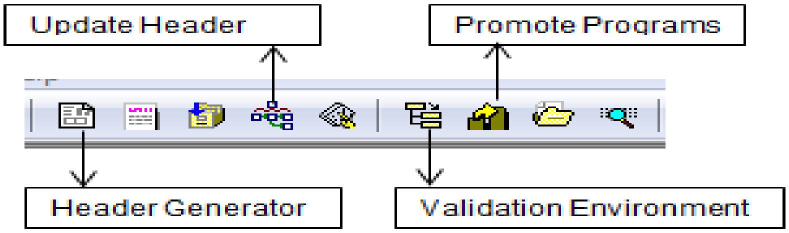
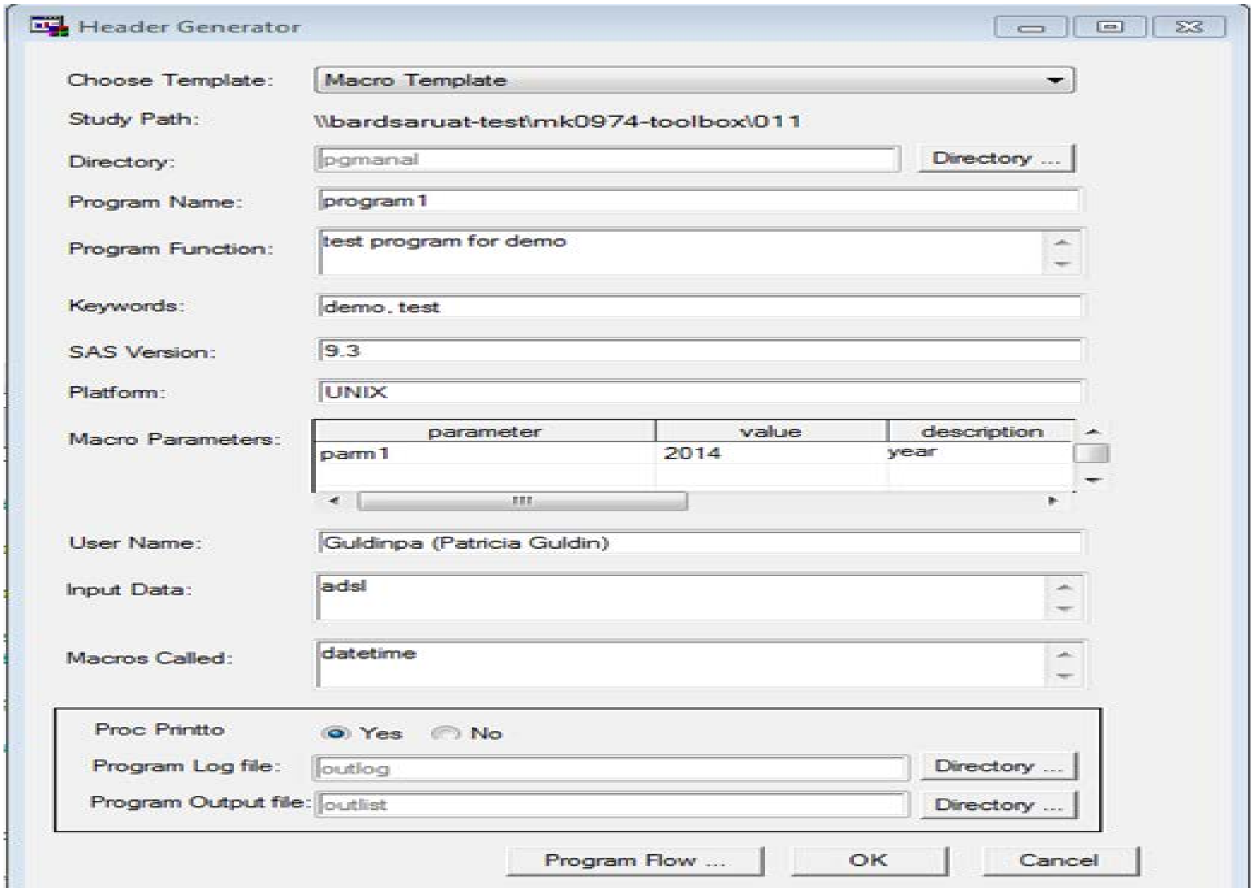


Figure 1: PC SAS toolbox function icons example

STANDARD HEADER GENERATOR

Upon selecting the Create Header function a programmer is presented with a screen and selects the name of the header template that they need. These templates are stored in a read only shared area and are maintained by the process owners. There are separate templates for programs and macros but each of the templates contains fields required by programming SOPs. SAS version, platform, program version date, programmer name and programmer id are pre-populated for the programmer from the catalog files. The programmer must enter additional information themselves. Examples include program name and description, any macro parameters to be used and input data. This is entered directly in the screen. Selections to add Proc Printto to the code or change output and log file location are available although log and output are defaulted to the locations defined by the directory structure and the value of protpath. A secondary screen is available that allows the programmer to enter and edit program flow steps to be entered into the program header. When the programmer is satisfied, they save and the program with the newly generated standard compliant header. The toolbox will not allow programmers to save programs in directories that start with “data” or “out” since these are defined for other purposes in the standard directory structure. This aids in ensuring adherence to the standard directory structure. The program is named with the name entered in the header program name field by the programmer. The programmer then continues with entering their SAS code. The Update Header section describes how the header is updated.



Choose Template: Macro Template
Study Path: \\bardsaruat-test\mk0974-toolbox\011
Directory: pgmanal
Program Name: program1
Program Function: test program for demo
Keywords: demo, test
SAS Version: 9.3
Platform: UNIX
Macro Parameters:

parameter	value	description
parm1	2014	year

User Name: Guldinpa (Patricia Guldin)
Input Data: adsl
Macros Called: datetime
Proc Printto ● Yes ○ No
Program Log file: outlog
Program Output file: outlist

Figure 2: Header Generator screen populated to create program1.sas

INSERT STEP

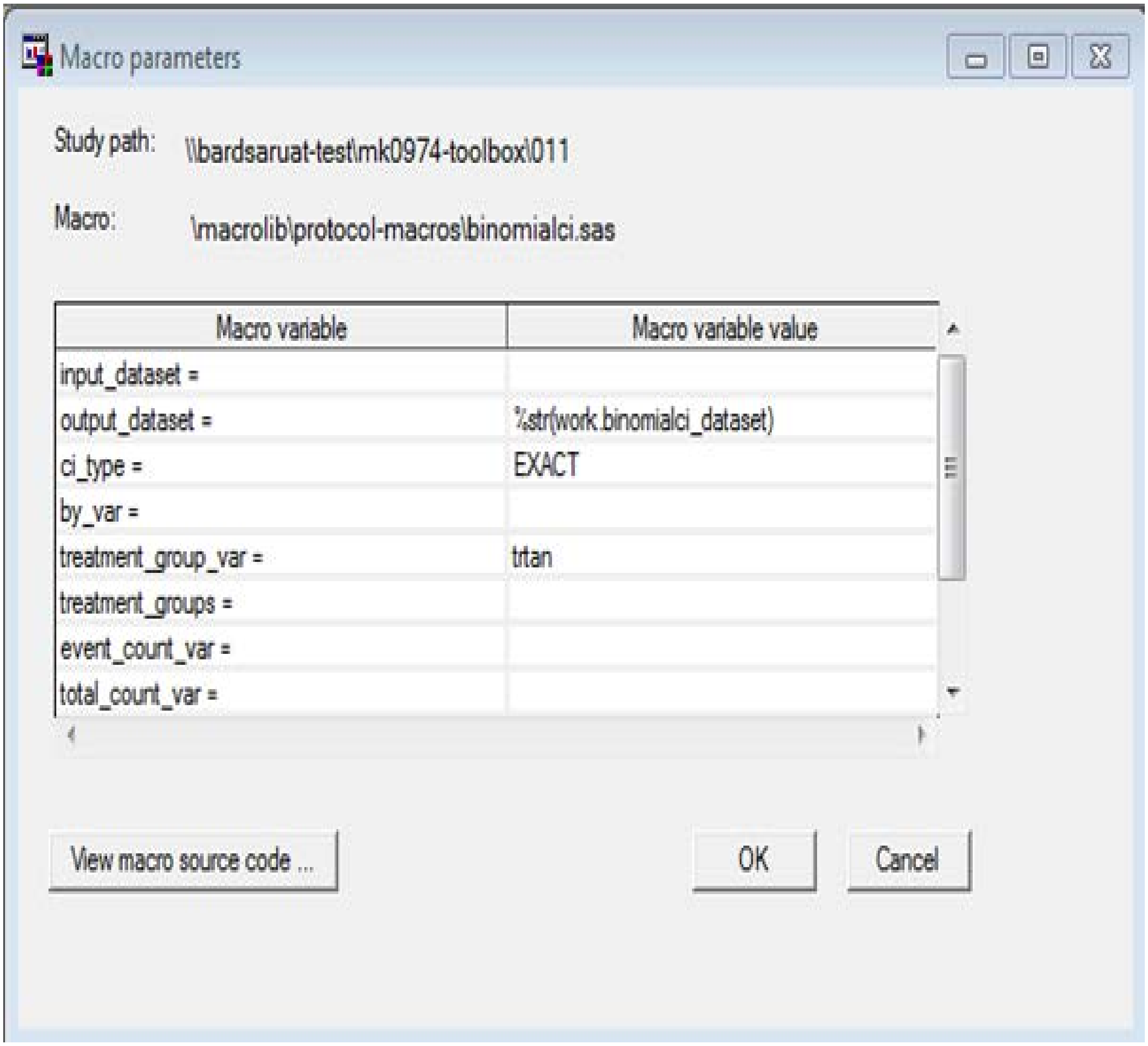
The Insert Step function is used to add steps to the program flow after initial creation of the header. Insert Step also adds commented sections in programmer defined locations in the code. Programmers position the cursor where they want the step description, select the Insert Step icon, and type the description text. The Update Header section describes how the inserted steps are numbered and synchronized with the program flow in the header.

PROC DATASETS DELETE

To insert code for Proc Dataset Delete a programmer positions the cursor where they want the code and selects the Proc Datasets Delete function. The names of the temporary data sets in the work library are listed in the delete statement. For proper functioning, temporary data sets must exist in the work library.

PICK UP MACRO

At times a programmer wants to add a call to a standard macro into their code. By placing their cursor at the desired insertion place and selecting the pick Up Macro function, a programmer is presented with available macro locations (defined by SASAUTOS) and a list of the macros in each location to choose from. Once a macro is selected the programmer can see all of the parameters and default values used in the selected macro and can modify the parameter values from this function. When the programmer confirms the selection, the macro call is inserted at the cursor location. The Update Header section describes how the header is updated to include this macro. In addition a read only copy of the macro source code can be displayed at the click of a button.



Study path: \\bardsaruat-test\mk0974-toolbox\011

Macro: \macrolib\protocol-macros\binomialci.sas

Macro variable	Macro variable value
input_dataset =	
output_dataset =	%str(work.binomialci_dataset)
ci_type =	EXACT
by_var =	
treatment_group_var =	trtan
treatment_groups =	
event_count_var =	
total_count_var =	

Figure 3: Pick Up Macro screen showing the variables used in the selected macro and the view macro source code button

UPDATE HEADER

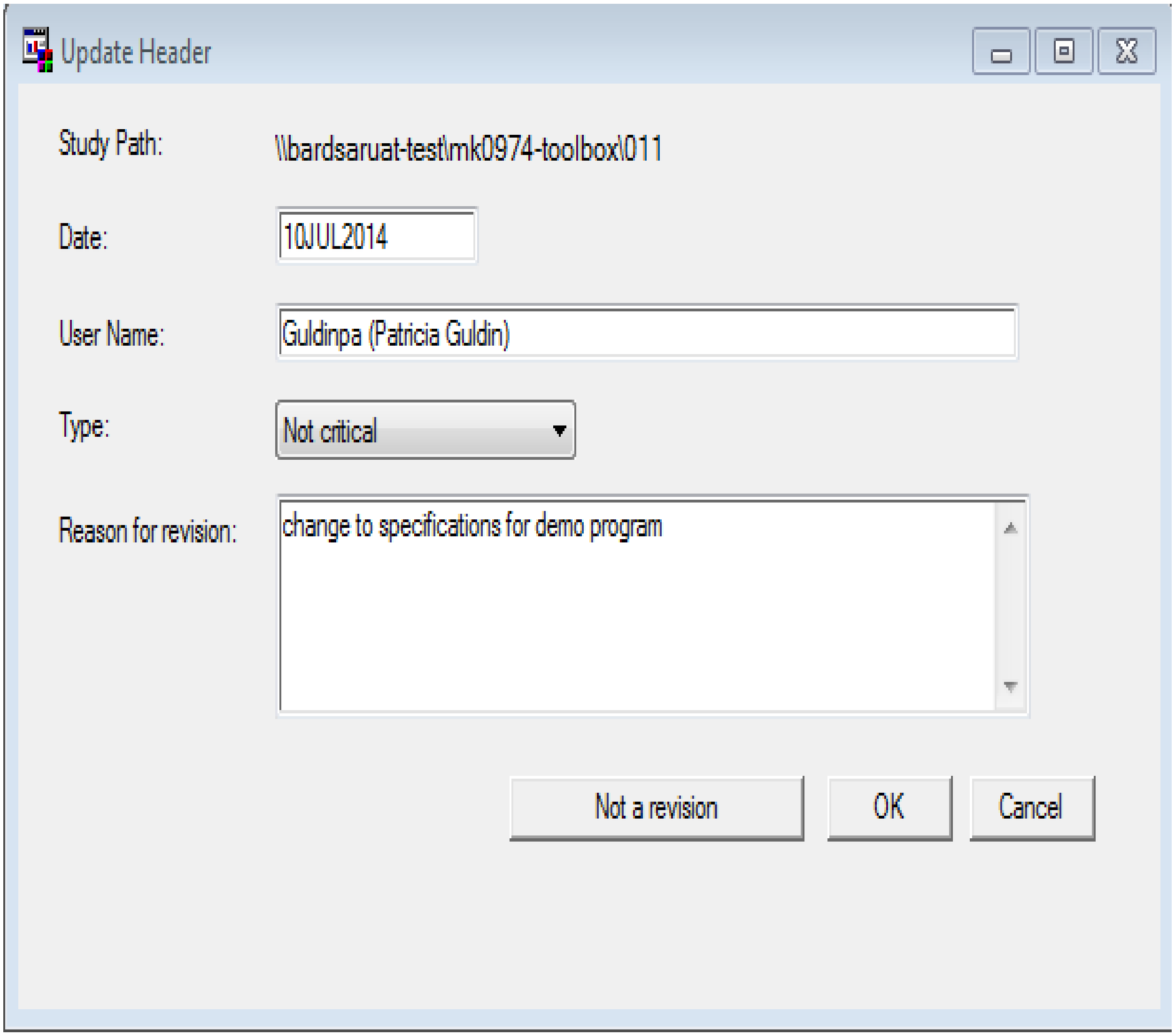
The Update Header function performs many important actions. Once all steps are inserted, all macros are added, and the code is ready, the programmer should use the update header function. This function first checks the header to see if it matches the standard compliant header template. If the header is not compliant, the programmer is prompted with an option to add a compliant header. The toolbox adds the fields from the compliant header template and retains the original header information so a programmer can copy information as needed into the appropriate places without having to re-type. Because the toolbox was designed to enable compliance, it will not process updates on programs with non-compliant headers. If a programmer chooses not to add the compliant header or manually modify the header so that it is compliant, they will need to make all header updates manually.

The Update Header function also updates inserted and modified steps in the program flow and commented sections of the code, renumbering the steps in order of their appearance. The commented section steps are used to populate and update the program flow section of the header.

Macro calls that were added manually or through the pick-up macro function are listed in the Macros called section of the header when the Update Header function is used.

The Update Header function updates the Version Date in the header with the current date. This ensures that programmers do not forget to change the version date in the header.

During the Update Header processing, the programmer is given the option to enter a reason for revision or indicate that the change is not a revision. If the update is a revision, the reason that the programmer enters along with the user id and date are added to the Revision History section of the header.



Study Path: \\bardsaruat-test\mk0974-toolbox\011

Date: 10JUL2014

User Name: Guldinpa (Patricia Guldin)

Type: Not critical

Reason for revision: change to specifications for demo program

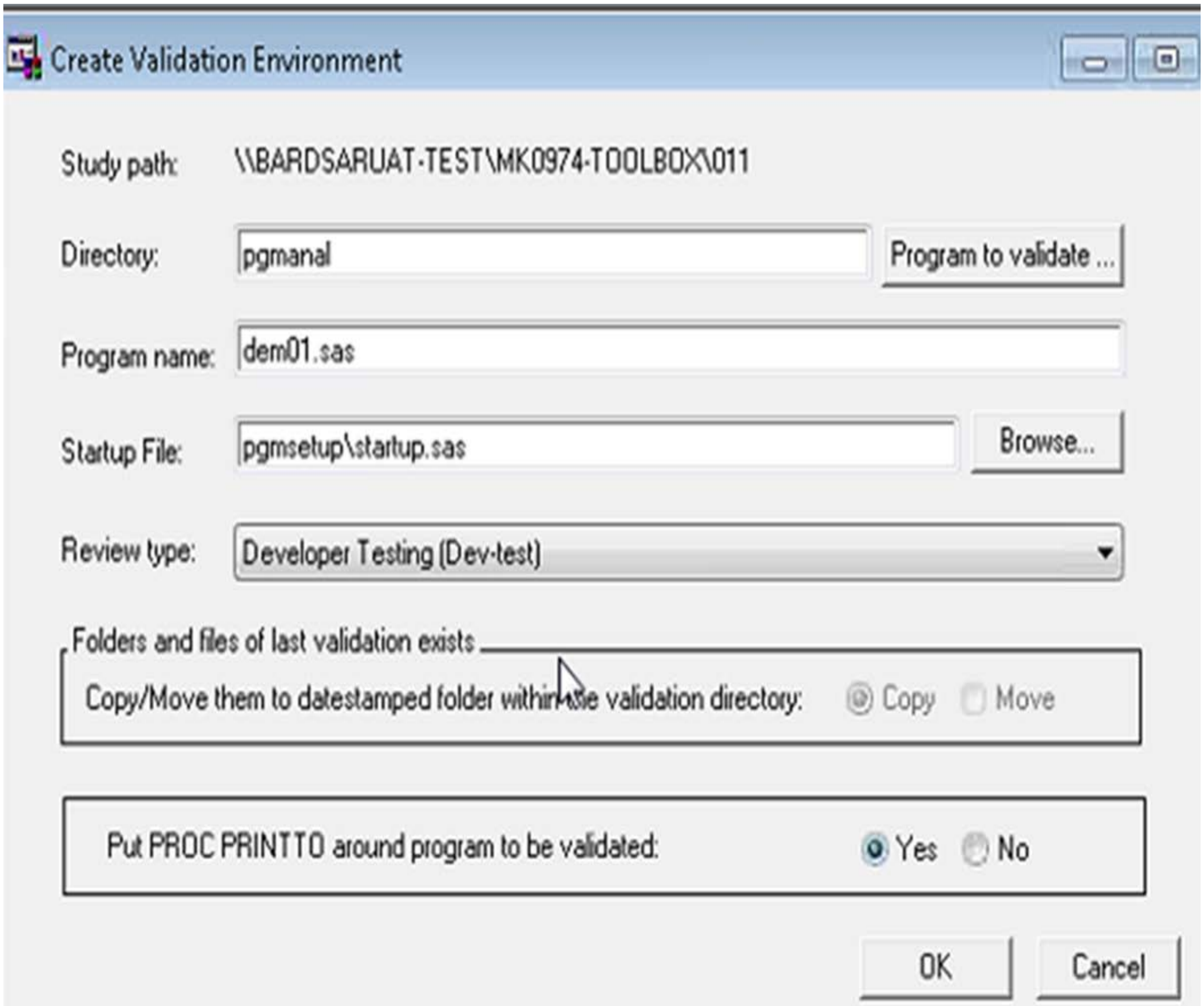
Figure 4: Update Header, user and date are auto-populated

CREATE VALIDATION ENVIRONMENT

When a programmer is ready to complete official validation of their code, they use the Create Validation function. From the Create Validation Environment screen they select the program they want to validate and the type of validation (Developer Testing, Double Programming, or Independent Validation). The Create Validation function creates the validation folder structure following the standard directory structure, adds a validation path macro variable to the startup program (to help direct output to the proper validation folders), copies the startup program to the validation folder, and opens the startup program for the programmer in SAS.

For the first execution of Developer Testing for a program, this function makes a copy of the program you want to validate and places it in the validate folder. It also creates a developer testing checklist, prepopulating the general information in the checklist such as project, program, programmer, file location and name and date. This function also saves the checklist in the checklist folder according to the standard directory structure and opens the checklist. For subsequent executions/rounds of Developer Testing for a program, this function will re-open the saved developer testing checklist for the programmer to edit so that only one version of the checklist is retained and maintained for each program.

For subsequent executions of any validation type for a program, this function provides the option to Copy or Move. Copy makes a backup of the previous execution of validation and saves it in a date time stamped folder. The original validation folders remain as they were for the last execution of validation. Copy could be useful if you have to modify one variable in your code and you want to modify and reuse your previous validation program to test. Move also makes a backup of the previous execution of validation and saves it in a date time stamped folder but the current validation folders are created and populated fresh, as they were the first time the programmer performed that validation type for that program. Move could be useful if you have to add a new variable that is not dependent on the current code, you don't need to re-test what you have already done but you do need to save proof of your previous testing.



Study path: \\BARDSARUAT-TEST\MK0974-TOOLBOX\011

Directory: pgmanal

Program name: dem01.sas

Startup File: pgmsetup\startup.sas

Review type: Developer Testing (Dev-test)

Folders and files of last validation exists

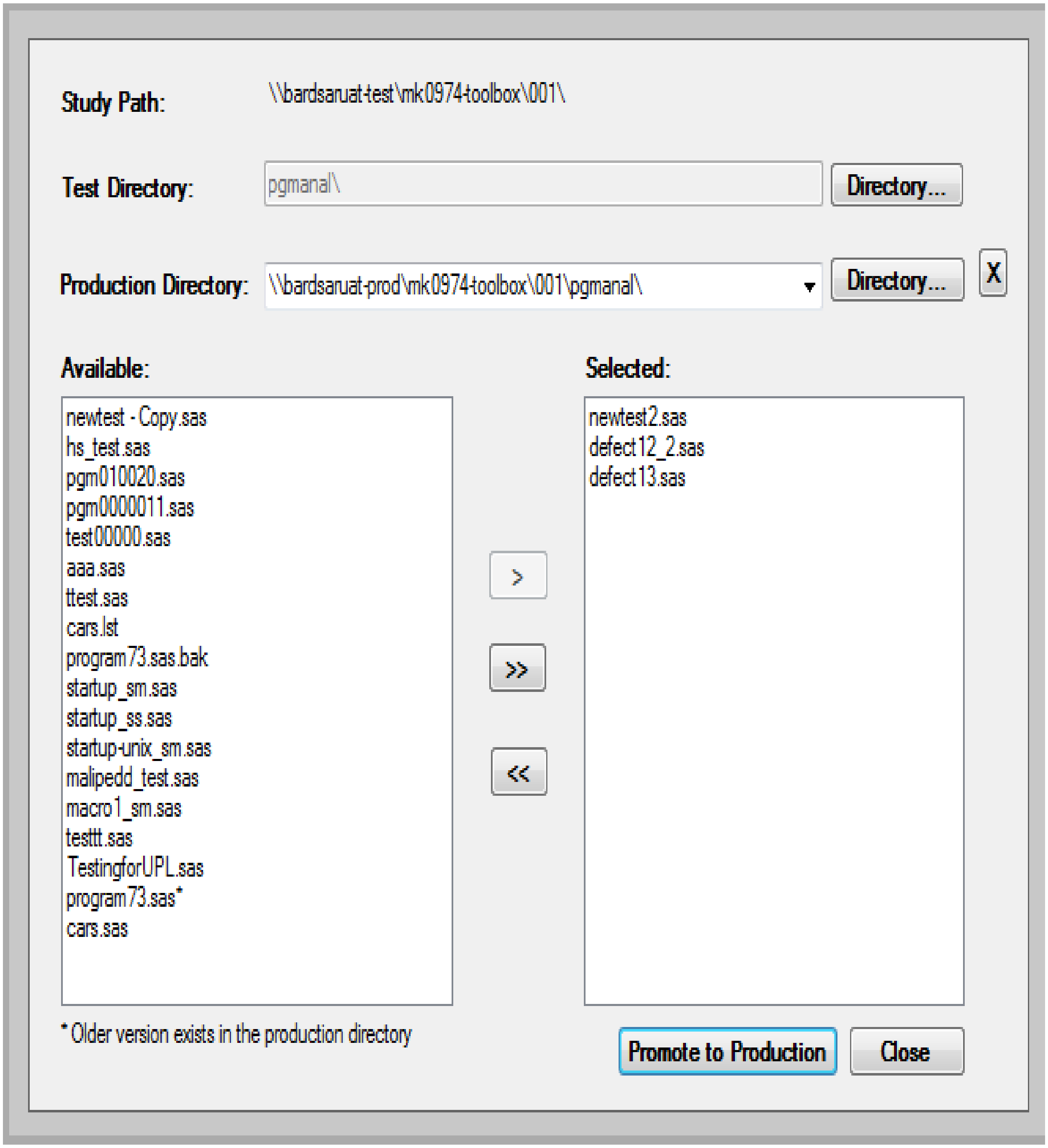
Copy/Move them to datestamped folder within the validation directory: ● Copy ○ Move

Put PROC PRINTTO around program to be validated: ● Yes ○ No

Figure 5: Create Validation Environment screen with Developer Testing selected

PROMOTE PROGRAMS

The Promote to Production function allows programmers to select one or many items to promote, or copy, from test to production. The production folders must exist prior to attempting this function and the programmer must have access to the test folders and access to write to the production folders. Promote Programs displays items that are available in a selected folder and will check if any items have already been promoted. If the function identifies items that already exist in production it will compare the dates to see if a newer version exists in test. Programs that were previously promoted but have newer versions in test are identified by an *. Newer versions of the same program overwrite the old versions in production when they are promoted to maintain one version of a program in production per the defined process. This function does not check that validation is complete and programmers are responsible for promoting items according to their validation plan and tracking.



Study Path: \\bardsaruat-test\mk0974-toolbox\001\

Test Directory: pgmanal\

Production Directory: \\bardsaruat-prod\mk0974-toolbox\001\pgmanal\

Available:

- newtest - Copy.sas
- hs_test.sas
- pgm010020.sas
- pgm0000011.sas
- test00000.sas
- aaa.sas
- ttest.sas
- cars.lst
- program73.sas.bak
- startup_sm.sas
- startup_ss.sas
- startup-unix_sm.sas
- malipedd_test.sas
- macro1_sm.sas
- testtt.sas
- TestingforUPL.sas
- program73.sas*
- cars.sas

Selected:

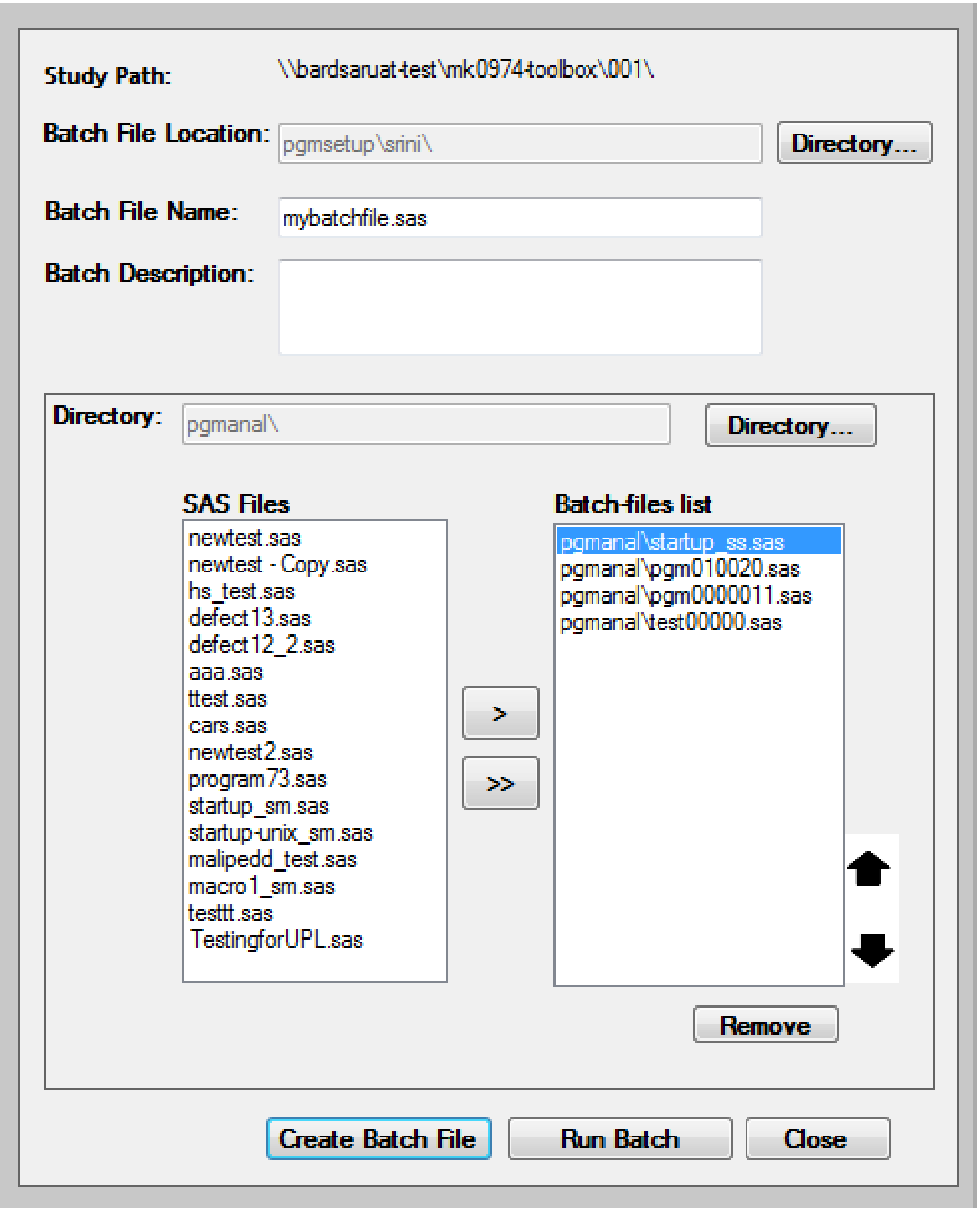
- newtest2.sas
- defect12_2.sas
- defect13.sas

* Older version exists in the production directory

Figure 6: Promote to Production screen, includes a program previously promoted with a newer version available in test

CREATE BATCH

The Create Batch function allows programmers to create and run batch jobs remotely in UNIX. The programmer selects the desired startup program and other programs to include in the batch. Programs from different folders can be included in a batch by navigating through the Directory button to those folders and selecting the desired programs until all programs for the batch have been selected. The programmer adds, removes and reorders the programs in the batch through this function using up and down arrows. The order of the programs in the Batch-files list determines the execution order. Batch jobs can be executed from this screen by selecting the Run Batch button. Batch jobs can also be saved for later use by selecting the Create Batch File button. The Create Batch function is useful for long running jobs as it will not tie up a programmer's SAS session and programmers are not required to be logged into UNIX for the job to execute.



Study Path: \\bardsaruat-test\mk0974-toolbox\001\

Batch File Location: pgmsetup\srini\

Batch File Name: mybatchfile.sas

Batch Description:

Directory: pgmanal\

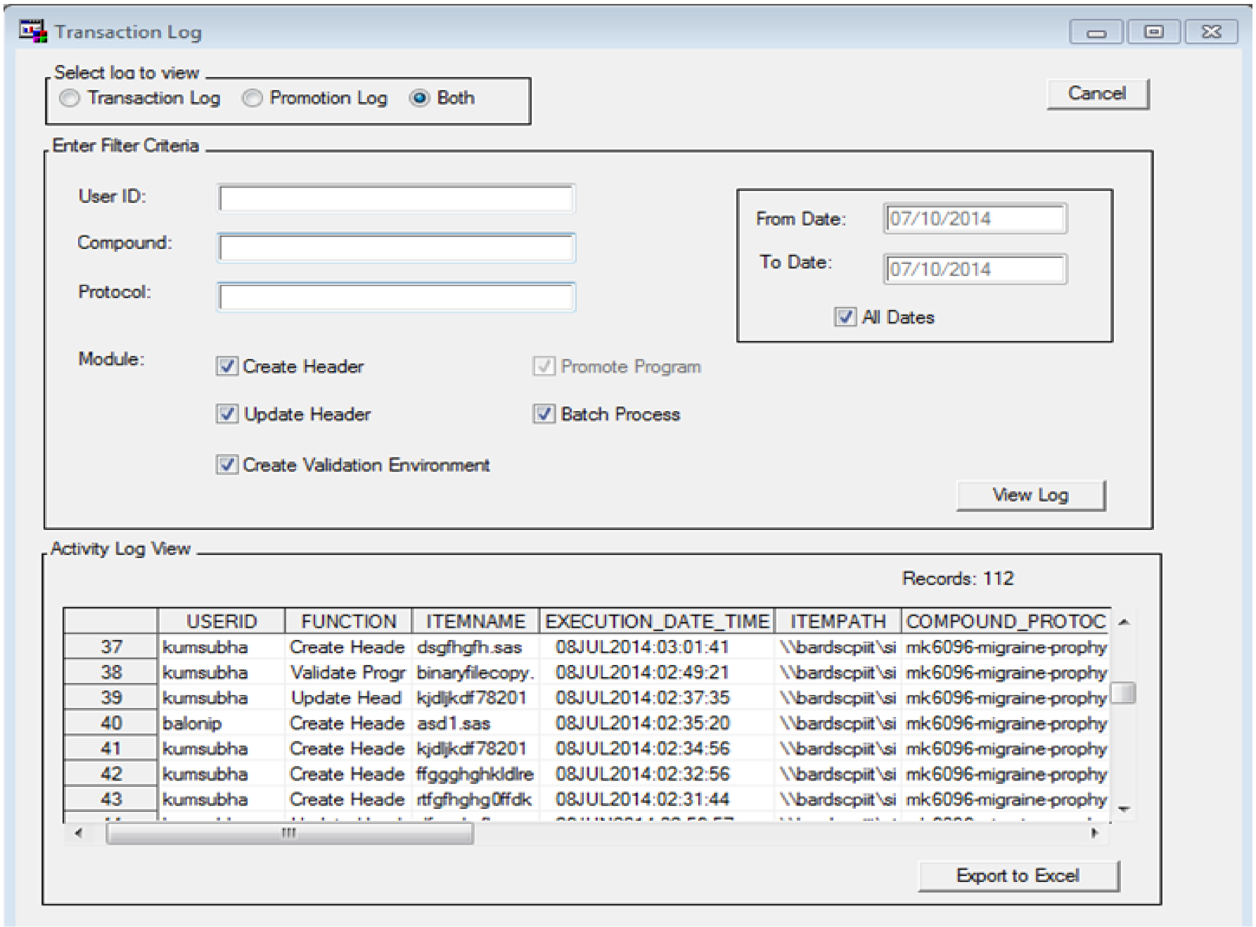
SAS Files
- newtest.sas
- newtest - Copy.sas
- hs_test.sas
- defect13.sas
- defect12_2.sas
- aaa.sas
- ttest.sas
- cars.sas
- newtest2.sas
- program73.sas
- startup_sm.sas
- startup-unix_sm.sas
- malipedd_test.sas
- macro1_sm.sas
- testtt.sas
- TestingforUPL.sas

Batch-files list
- pgmanal\startup_ss.sas
- pgmanal\pgm010020.sas
- pgmanal\pgm0000011.sas
- pgmanal\test00000.sas

Figure 7: Create batch job screen showing some programs selected

MANAGE TEMPLATES AND ACTIVITY LOG

There are two main administrative functions available in the toolbox. The ability to update templates and checklists is only available to admin users since changes to these require impact analysis to ensure proper functioning of the toolbox after they are introduced. The Transaction Log function is available to all programmers. The toolbox retains records of Create Validation Environment, Create Header, Update Header, Promote Programs, and Create Batch. These records include user ids and dates. Programmers can filter for the information they want to see and can view the results onscreen or export them to excel. This type of information may be useful to check compliance, or to see how many modifications were made to a program for example.



Select log to view: ○ Transaction Log ○ Promotion Log ● Both

Enter Filter Criteria

User ID:

Compound:

Protocol:

From Date: 07/10/2014

To Date: 07/10/2014

☑ All Dates

Module: ☑ Create Header ☑ Promote Program ☑ Update Header ☑ Batch Process ☑ Create Validation Environment

Activity Log View

Records: 112

	USERID	FUNCTION	ITEMNAME	EXECUTION_DATE_TIME	ITEMPATH	COMPOUND_PROTOC
37	kumsubha	Create Heade	dsgfhgfh.sas	08JUL2014:03:01:41	\\bardscpiit\si	mk6096-migraine-prophy
38	kumsubha	Validate Progr	binaryfilecopy.	08JUL2014:02:49:21	\\bardscpiit\si	mk6096-migraine-prophy
39	kumsubha	Update Head	kjdljkdf78201	08JUL2014:02:37:35	\\bardscpiit\si	mk6096-migraine-prophy
40	balonip	Create Heade	asd1.sas	08JUL2014:02:35:20	\\bardscpiit\si	mk6096-migraine-prophy
41	kumsubha	Create Heade	kjdljkdf78201	08JUL2014:02:34:56	\\bardscpiit\si	mk6096-migraine-prophy
42	kumsubha	Create Heade	ffggghghkldlre	08JUL2014:02:32:56	\\bardscpiit\si	mk6096-migraine-prophy
43	kumsubha	Create Heade	rtfgfhghg0ffdk	08JUL2014:02:31:44	\\bardscpiit\si	mk6096-migraine-prophy

Figure 8: Transaction Log showing available filters and sample output

CONCLUSION

With the first release of the programming compliance toolbox there have been notable gains. The automation and auto-population that the toolbox provides allows programmers to more efficiently be compliant. Programmers can spend less time on documentation and process requirements and focus their time on the specialty of writing code. Header information and updates are automated and much of the data entry is done by the toolbox. Remembering where to save things is no longer a concern since the toolbox knows where things belong and saves them automatically. Programmers no longer need to perform manual tasks such as creating validation folders and checklists since these tasks are performed by the toolbox. The toolbox is easy to use with buttons to click, drop down lists provided when information is not auto-populated, and some built in compliance checking. The toolbox helps programmers globally to more consistently follow the SOPs and processes which are aligned with agency regulations and company policies. Because the toolbox is designed to work with the standardized computing platform, it enables resources globally to be confident in the location and content they will find for any project, thus saving time when re-allocation of resources is required to meet timelines.

As more data is compiled by the transaction log it will be examined to determine what reports can be generated to display process or training gaps, compliance gains or issues, to help research audit findings and check remediation, or to be used in employee performance assessments. Perhaps the transaction log information could be provided as supporting evidence that programming processes were followed.

The toolbox has helped increase awareness of the importance of the departmental SOPs and processes, has sparked conversations about process improvement, increased individual accountability, and fostered a culture of compliance. It is expected that future releases of the toolbox can address enhancements and any process updates that come out of process improvement ideas.

REFERENCES

Coppin, Frederic and Herremans, Carl. 2003. “A standard SAS programming toolbar: A step forward for GPP/SOP compliant SAS program development.” *Proceedings of the PharmaSUG 2003 Conference*. Available at http://www.lexjansen.com/pharmasug/2003/ApplicationsDevelopment/ad041.pdf.

ACKNOWLEDGMENTS

I would like to thank Carl Herremans for his development of the first toolbar and his input on the specifications for the programming compliance toolbox.

CONTACT INFORMATION

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

Name: Patricia Guldin
Enterprise: Merck & Co., Inc., Kenilworth, NJ USA
Address:
City, State ZIP: Upper Gwynedd, PA
Work Phone: 267-305-8242
E-mail: patricia_guldin@merck.com