PharmaSUG 2013 - Paper BB13

Avoiding SAS Data Set Locks in a Windows Environment

Brandon Graham, PPD, Wilmington, NC
Scott Osowski, PPD, Wilmington, NC

ABSTRACT

If you have ever worked in a Windows environment where SAS® data sets were stored in a common folder accessed by multiple users, you have likely encountered the frustrating situation of SAS data sets being locked by another user. This is a common occurrence when a reviewer or another programmer has opened the data set from the shared directory for viewing. This paper details a solution to avoid SAS data set locks through SAS and Windows registry modifications which alter the way SAS data sets are opened for viewing and help ensure the source data sets are not locked for other users.

INTRODUCTION

The need for multiple users to access SAS data sets from the same Windows directory is common in many work environments. Issues can sometimes arise when more than one user needs to view or access the same SAS data set simultaneously. SAS offers a solution to the simultaneous access problem with the SAS/SHARE® product which allows SAS data sets to be updated simultaneously by multiple users. However, not all organizations have the need for the full functionality of SAS/SHARE or have not made it a standard for their working environment. In such cases users must be careful to access shared SAS data sets in a way that does not interrupt the work of others. In this paper we will look at the most common cause of SAS data set locks and offer a technical solution to help avoid this problem. We will also discuss some basics of the SAS and Windows registry that you need to understand to implement this solution.

WHY DO SAS DATA SET LOCKS OCCUR?

In a Windows environment SAS data sets are locked for other users when a SAS program attempts to write to a SAS data set or when a user has a data set open for viewing.

If you work in an environment in which multiple users or programs have the need to write to a single SAS data set simultaneously, you will need a solution such as SAS/SHARE. In our experience in the clinical research industry, this need to have multiple programs update a data set simultaneously is uncommon.

A more common way SAS data sets are locked in this industry occurs when a user has the data set opened for viewing. In this situation one team member is reviewing the SAS data set directly from the shared location, and another team member is attempting to update the SAS data set programmatically. The data set will be locked for others regardless if it is opened directly from the Windows folder, or if it is opened from a LIBNAME which is established within a SAS session to reference the shared folder. The longer the SAS data is opened for viewing, the more likely a potential conflict arises from another user trying to write to the data set. To make matters worse, it is often difficult to determine which user has the data set lock without administrator access to the server where the data resides.

In the remainder of this paper, we will discuss a technical solution which is designed to avoid the problem of SAS data sets being locked while opened for viewing. The solution is fairly simple conceptually. Rather than opening a data set for viewing directly from the shared folder, we want to ensure that the data is first copied to the user's WORK library and then opened for viewing. We also want the solution to be done automatically and in a way that is nearly transparent to users.

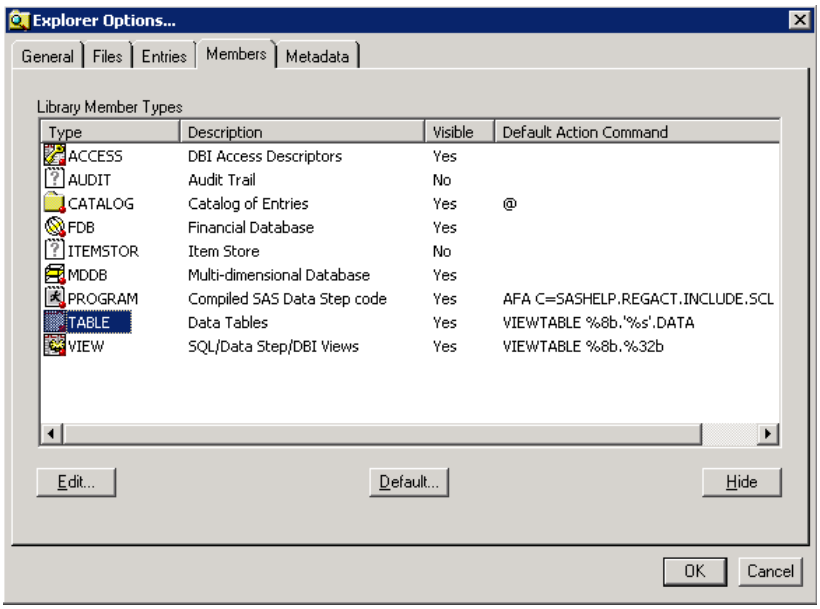
Note: The following discussion and proposed solutions assume some knowledge of the SAS and Windows registries. References for further reading have been listed in the Recommended Reading section towards the end of this paper. We recommend that only experienced users make modifications to registry values. It is always a good idea to make a backup of the registry prior to making modifications. In addition, there may be differences between SAS installations that make the solution slightly different from environment to environment.

AVOIDING LOCKS WITHIN A SAS DISPLAY MANAGER SESSION

SAS offers users a lot of flexibility for customizing the SAS® Display Manager. Users can adjust color, default window size, toolbars, and many other options. Modifications can also be made to the default action for file types when

accessed through the SAS session. The ability to modify the default action for SAS data sets is the key to avoiding locks when users open a SAS data set for viewing from within an interactive SAS session.

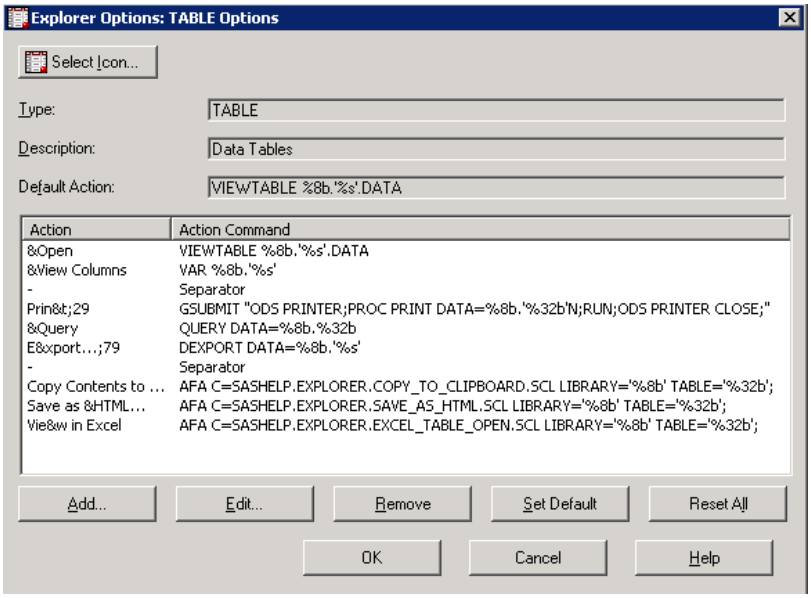
Within the Display Manager, open the SAS Explorer (select Tools > Options > Explorer). On the Members tab, we see several SAS library member types (or file types) and the default action for each member type. Here we have the ability to modify the actions that SAS associates with each file type.



Display 1. SAS Explorer Members Tab

The TABLE member controls the actions associated with SAS data sets, including what actions occur when a user opens a SAS data set for viewing within a SAS session.

Select TABLE and then Edit to modify the actions associated with SAS data sets.

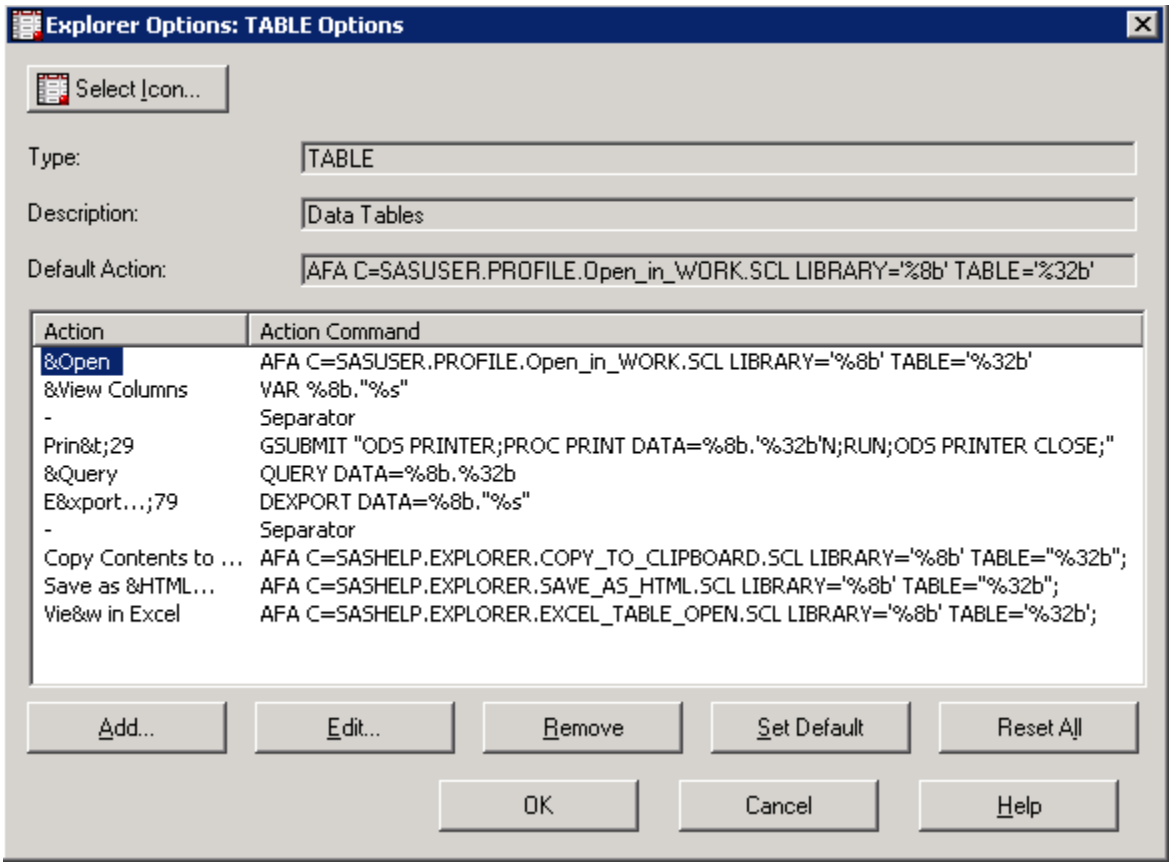


Display 2. SAS Explorer Table Options (Default)

The actions associated with a SAS data set are shown above in Display 2. You will notice that each line is associated with a menu item displayed when you right-click on a SAS data set within a SAS session. You have the ability to modify these actions and to add additional actions within this TABLE Options screen.

The Open action by default will open a SAS data set for viewing. The data set selected is opened directly from its current location, which is referenced by a LIBNAME defined within the SAS session. As we discussed, viewing the data in this way locks the data for any other users trying to modify the data.

Since we can modify the action command for each action, we can instruct SAS to first copy the data set to the WORK library, and then open the data for viewing. We will do this by modifying the Action Command for the Open Action. Rather than open the SAS data set directly, instead we want to direct SAS to run an SCL program which will open the data set in our preferred method.



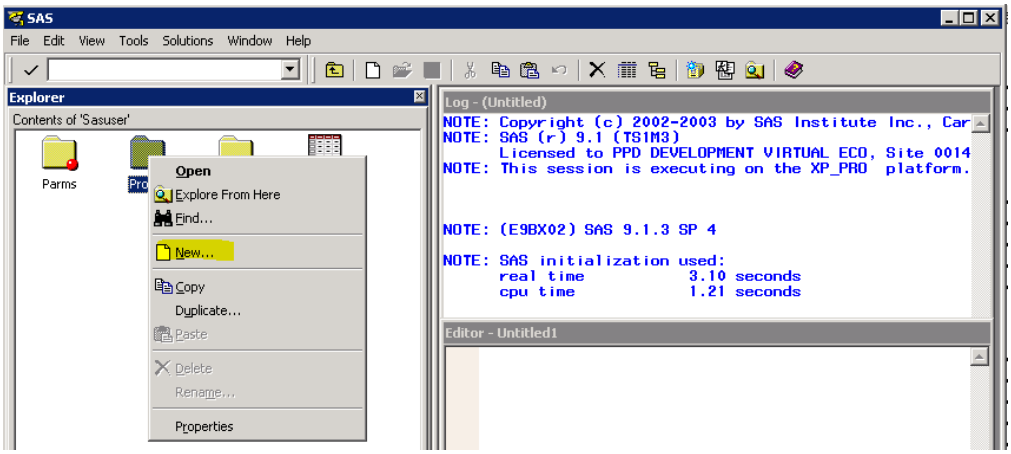
Display 3. SAS Explorer Table Options (With Modification to &Open Action)

In Display 3 above, we have changed the Open action command to direct SAS to run an SCL program named "Open_in_Work.SCL" which has been saved in the SASUSER.PROFILE area. Also notice that we have ensured that the Default Action for the Table member is to execute this SCL program (this instructs SAS to take this action when a SAS data set is double-clicked within the Display Manager session). *Note: If you prefer not to change the default "Open" action, consider instead adding a new action such as "Open in Work" and setting this new action as the default.*

The "Open_in_Work.SCL" code is displayed in Appendix A. This code first determines the location and data set name to be viewed. If the data set to be opened is located in the WORK library, then it is opened as usual. If the data to be opened is located somewhere other than the WORK library, then it is first copied to the WORK library and opened from there. This SCL code also contains a check to determine if a data set already exists in the WORK library. If the data set already exists in WORK, the data set being opened is renamed by adding the date/time as a suffix to the data set name. This ensures that we do not unintentionally overwrite any existing WORK data set.

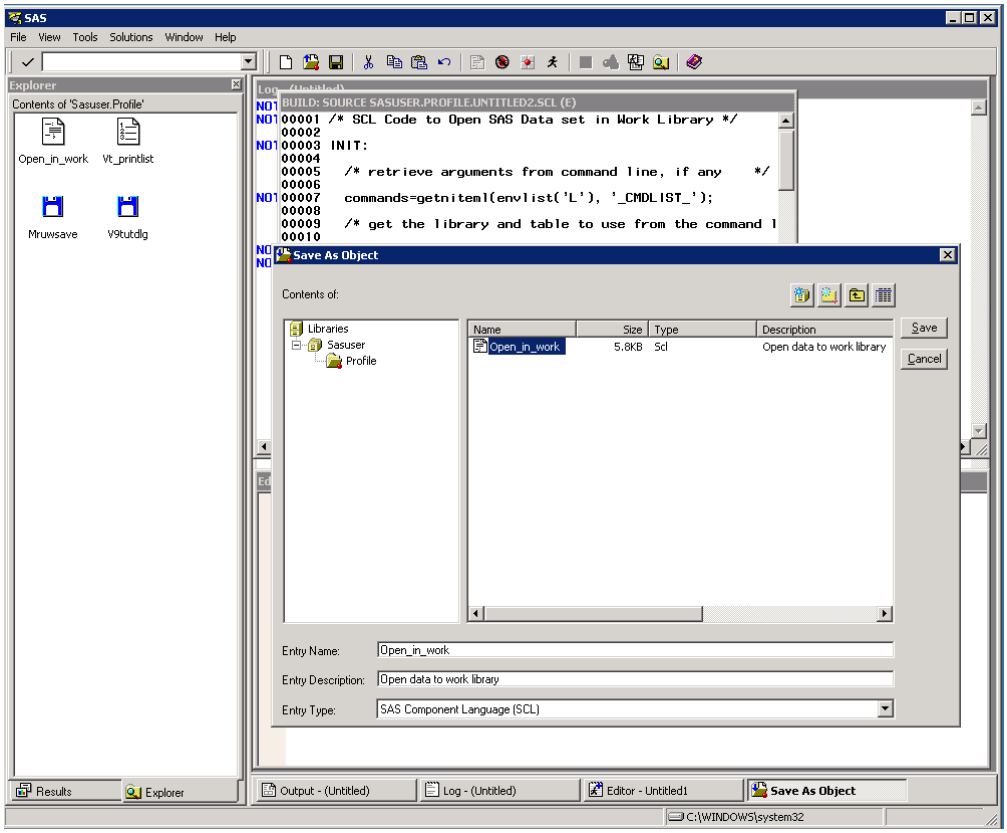
How to Compile SCL Code in SASUSER.PROFILE

To save and compile the SCL code in the SASUSER.PROFILE area, first open a Display Manager session. Within SAS Explorer go to Libraries>SASUSER, right click on “Profile” and select “New”:



Display 4. Creating a New Entry in SASUSER.PROFILE

Once the New Entry in SASUSER.PROFILE window opens, select “SCL Program” and “Ok”. In the Build Window, copy the code from Appendix A, paste it into the build editor and use the Save button to save the code as “Open_in_work”:



Display 5. Adding SCL Program to SASUSER.PROFILE

Now in the Profile folder, you will see the “Open_in_work” object; simply right-click and select “Compile”. You have now successfully saved and compiled the SCL code to your SASUSER profile. Please note the modifications

described here are SAS user and version specific. Thus, if you use multiple versions of SAS you will need to perform this step as well as the SAS Explorer table modification to the "&Open" action once for each version of SAS you use.

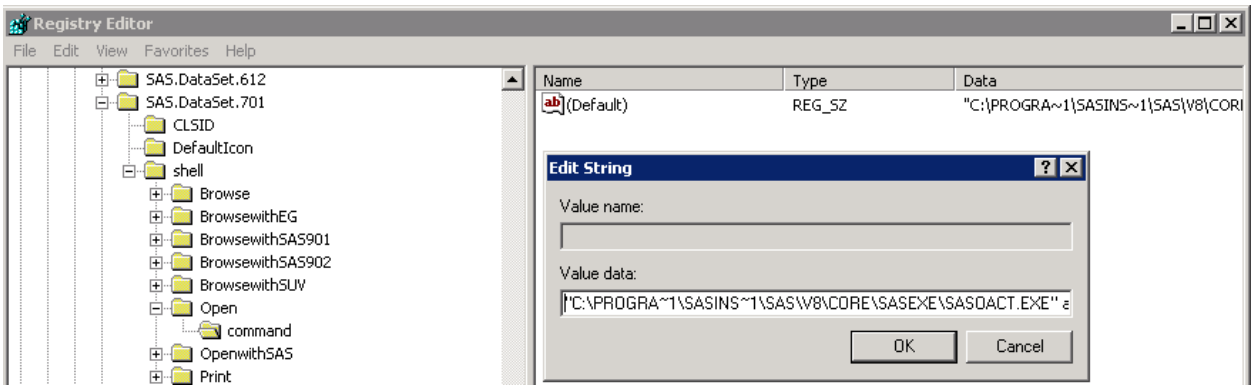
The modifications described above will ensure that SAS data sets opened for viewing within a Display Manager session are opened from the user WORK library and these data sets are not locked for other users.

AVOIDING SAS DATA SET LOCKS WHEN OPENED VIA WINDOWS

Even with the modifications described above within the Display Manager session in place, it is still possible for SAS data sets to be locked when someone views a data set directly from a shared Windows location. How do we ensure that viewing data in this way does not lock the data for other users? For that, we need to expand beyond Display Manager settings and examine what is happening in the Windows registry when a user opens a SAS data set.

What happens when you open a SAS data set for viewing directly from a Windows folder? The answer to this question lies within the Windows registry. You can determine this by browsing through the Windows registry using the Registry Editor (regedit.exe) and finding the command value which is associated with the SAS data set file type. You will likely find that the command issued is to the SASOACT.EXE file (the SAS OLE automation server). The exact command in our environment is as follows:

```
"C:\PROGRA~1\SASINS~1\SAS\V8\CORE\SASEXE\SASOACT.EXE" action=Open filename="%1"
progid=SAS.Application.8
```



Display 6. Windows Registry Value Associated with Opening a SAS Data Set

When a user opens a SAS data set directly from a Windows folder location, this command is sent to the SAS OLE automation server (SASOACT.EXE). The SAS OLE automation server first determines if you have any running SAS sessions. If a running SAS session is found, then the data set being opened for viewing is opened within an existing SAS session. If no running SAS sessions exist, then a new SAS session is started and the data set is opened within the new session. In either case, the data set is opened directly from its current location, so it is locked for any other users who attempt to update the data for the duration the data set is opened for viewing.

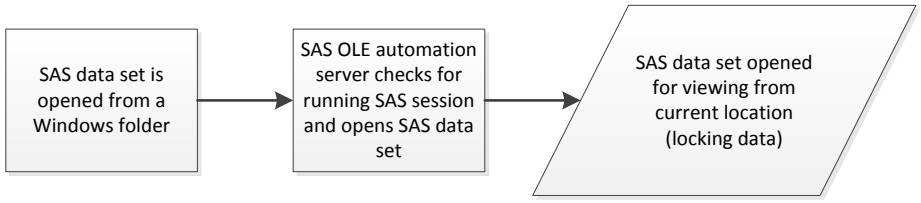


Figure 2. Default Process for Opening Data Sets from Windows Command

Once the SAS OLE automation server receives the command to open a SAS data set from a shared location, we are not aware of any modifications which would avoid a data lock. However, there is a workaround. Instead of passing a command to SASOACT.EXE to open a SAS data set, we could instead pass an SAS program to SASOACT.EXE for execution which would first copy the SAS data set to the WORK library prior to opening. Figure 2 below shows this modified process flow.

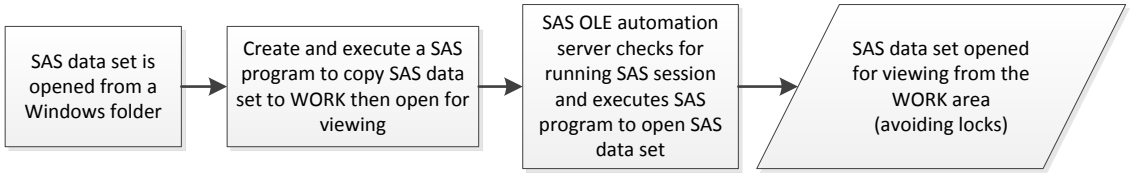


Figure 2. Modified Process for Opening Data Sets from Windows Command

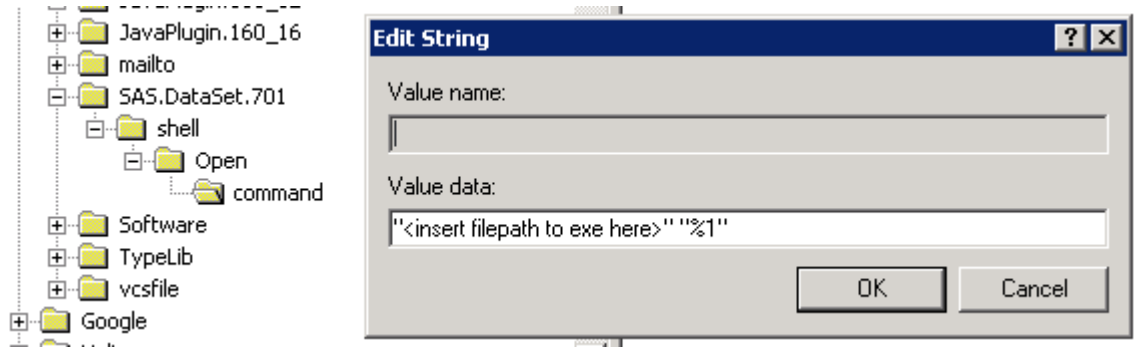
Let's take a look at the specifics of how this modification can be accomplished.

First, we will need to create a script or an executable (.EXE) which will serve as this intermediate step between the Windows command line and the SAS OLE automation server. This can be accomplished in a variety of languages (we chose VB.NET due to familiarity and ease of use with the Windows environment), but it needs to accomplish the following:

- parse the command line to determine the name and location of the SAS data set which was selected for opening within the Windows environment
- produce an SAS program which will first copy the data set to WORK and then open for viewing
- submit this SAS program as a command to the SAS OLE automation server (SASOACT.EXE)
- remove the SAS program (clean up after itself)

An example of this program (written in VB.NET) is shown in Appendix B.

Next, we need to change the Windows registry command for SAS data sets to call this intermediate executable instead of calling the SAS OLE automation server directly. This is accomplished by modifying the Windows registry command for SAS data sets.



Display 7. Example Modification to Registry Value Associated with Opening a SAS Data Set

The "%1" value referenced in the "Value data:" line above passes the file path of the data set selected for opening within the Windows environment. This value can then be parsed and used in the SAS program creation. The resulting SAS program is then sent to the SAS OLE automation server for execution. The SAS OLE automation server then takes over to execute the program which will open the data in a Display Manager session in a way that it avoids locking the data for other users by first copying the data set to the WORK library.

CONCLUSION

We have examined two aspects of a technical solution involving SAS and Windows registry modifications which can be implemented in a work environment to help avoid SAS data set locks and the resulting frustration and loss of productivity. We have focused on data set locks which specifically occur when data sets are opened by the Display Manager (either from within the Display Manager or from a Windows folder). We have not discussed data set locks which can result when data sets are opened for viewing by other applications such as the SAS® System Viewer or SAS® Universal Viewer. Perhaps this will be a topic for a future paper. Even if the modifications described here are not appropriate for your environment, the concepts and customizations discussed show some of the power that you have to modify default actions within SAS and Windows.

ACKNOWLEDGMENTS

A special thanks to Tom Fritchey for his technical expertise and guidance which helped to make this paper possible.

RECOMMENDED READING

- SAS 9.2 Language Reference – The SAS Registry: http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a002261741.htm
- "SAS Explorer: Use and Customization", Richard A. DeVenezia: http://www.nesug.org/proceedings/nesug05/ap/ap6.pdf
- "Doing More with the SAS® Display Manager: From Editor to ViewTable - Options and Tools You Should Know", Arthur L. Carpenter: http://support.sas.com/resources/papers/proceedings12/151-2012.pdf
- Microsoft Support – How to Modify the Windows Registry: http://support.microsoft.com/kb/136393

CONTACT INFORMATION

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

Brandon Graham
PPD
929 N. Front St.
Wilmington, NC 28401
Work Phone: +1 (910) 558-7386
Fax: +1 (919) 654 5505
E-mail: brandon.graham@ppdi.com

Scott Osowski
PPD
929 N. Front St.
Wilmington, NC 28401
Work Phone: +1 (910) 558-7546
Fax: +1 (919) 654 5505
E-mail: scott.osowski@ppdi.com

APPENDIX A.

SCL program which was saved to SASUSER.PROFILE.

```
/* SCL Code to Open SAS Data set in Work Library */

INIT:

  /* retrieve arguments from command line, if any */

  commands=getniteml(envlist('L'), '_CMDLIST_');

  /* get the library and table to use from the command line */

  library=getnitemc(commands,'LIBRARY',1,1,'');
  table=getnitemc(commands,'TABLE',1,1,'');

  /* verify that we got a table specification */

  if table='' or library='' then do;
    message=makelist();
    temp='A table must be specified.';
    message=insertc(message,temp,-1);
    rc=messagebox(message,'S','O',"Input Error");
    message=clearlist(message);
    rc=rc;
    return;

  end;

  /* Now let's build up a submit block and submit the    */
  /* code necessary to open data set in the work library */

submit immediate;

 %macro copy_to_work();
 /* If data set being opened from WORK, open as usual */
 %if %upcase(&library)=WORK %then %do;
   dm 'viewtable data=work.&table' viewtable;
 %end;
 /* If data being opened elsewhere, copy to WORK */
 %else %do;
  /* First check if the data set already exists in WORK */
  %if %sysfunc(exist(work.&table)) %then %do;

    /* If data set already exists in WORK, rename the   */
    /* data set based on the current date/time and copy */
    /* to work (to avoid overwriting existing WORK data */
    data _null_;
    call symput('mydate', compress(put(date(), yymmdd10.),'-'));
    call symput('mytime', trim(left(compress(put(time(), time9.),':'))));
    run;

    data &table&mydate._&mytime.;
     set &library.&table;
    run;

    /* Open the renamed data set for viewing */
    dm 'viewtable data=work.&table&mydate._&mytime.' viewtable;

   %put Note: &library.&table renamed to &table&mydate._&mytime. and opened in WORK library;
  %end;
  %else %do;
    /* If data does not already exist in work, copy data directly without rename */
    proc copy in=&library out=work;
      select &table;
    run;

    /* Open the data set for viewing */
```

```
    dm 'viewtable data=work.&table' viewtable;

   %put Note: &library.&table data set copied and opened in WORK library;
  %end;
%end;
%mend copy_to_work;

%copy_to_work()

endsubmit;

RETURN;
```

APPENDIX B.

VB.NET code to create a SAS program to open data sets in WORK and submit the program to the SAS OLE Automation Server for execution. This code was created for the .NET Framework 3.5.

```
Imports System.IO
Imports System.Text

Module Module1
  Sub Main()

  Try
   Dim SASDataSetFilePath As String = My.Application.CommandLineArgs(0).ToString.Trim
   Dim SASDataSetDirectory As String = Path.GetDirectoryName(SASDataSetFilePath).Trim
   Dim SASDataSetName As String = _
      Path.GetFileNameWithoutExtension(SASDataSetFilePath).Trim

   ' write SAS program to copy to data to WORK library and then open for viewing
   Dim tempSASFileName As String = System.IO.Path.GetTempFileName.Replace(".tmp", _
     ".sas")
   Dim sb As New StringBuilder

   sb.AppendLine("libname myopnlib '" & SASDataSetDirectory & "' access=readonly;")
   sb.AppendLine()
   sb.AppendLine("%macro open_in_work();")
   sb.AppendLine("%if %sysfunc(exist(" & SASDataSetName & ")) %then %do;")
   sb.AppendLine("data _null_;")
   sb.AppendLine("call symput('mydate', compress(put(date(), yymmdd10.),'-'));")
   sb.AppendLine("call symput('mytime', trim(left(compress(put(time(), _
       time9.),':'))));")
   sb.AppendLine("run;")
   sb.AppendLine()
   sb.AppendLine("data " & SASDataSetName & "&mydate._&mytime.;")
   sb.AppendLine(" set myopnlib." & SASDataSetName & ";")
   sb.AppendLine("run;")
   sb.AppendLine()
   sb.AppendLine("dm ""viewtable data=work." & SASDataSetName & "&mydate._&mytime."" _
        viewtable;")
   sb.AppendLine()
   sb.AppendLine("%put " & SASDataSetName & " renamed and opened from WORK;")
   sb.AppendLine("%end;")
   sb.AppendLine("%else %do;")
   sb.AppendLine("proc copy in=myopnlib out=work;")
   sb.AppendLine(" select " & SASDataSetName & ";")
   sb.AppendLine("run;")
   sb.AppendLine()
   sb.AppendLine("dm 'viewtable data=work." & SASDataSetName & "' viewtable;")
   sb.AppendLine("%end;")
   sb.AppendLine("libname myopnlib clear;")
```

```
    sb.AppendLine("%mend open_in_work;")
    sb.AppendLine("%open_in_work()")

    Using outfile As New StreamWriter(tempSASFileName)
      outfile.Write(sb.ToString())
    End Using

    ' pass the SAS program to SASOACT
    Process.Start("C:\PROGRA~1\SAS9~1.2\SASFOU~1\9.2\CORE\SASEXE\SASOACT.EXE", _
       "action=Submit datatype=SASFile filename=""" & tempSASFileName & """ _
        progid=SAS.Application.902")

    ' wait for SAS to find program, then delete temporary SAS program from Temp folder
    Do
     Threading.Thread.Sleep(5000)
     If File.Exists(tempSASFileName) Then File.Delete(tempSASFileName)
     If File.Exists(tempSASFileName.Replace(".sas", ".tmp")) Then
       File.Delete(tempSASFileName.Replace(".sas", ".tmp"))
    Loop While File.Exists(tempSASFileName)

    Catch
  End Try
End Sub

End Module
```