PharmaSUG 2015 - Paper QT42

Automating Clinical Trial Reports with ODS ExcelXP Tagset

John O'Leary, Department of Veterans Affairs, West Haven, CT

ABSTRACT

This paper offers a solution for generating multiple Excel workbooks by combining SAS® Base tools and a Visual Basic for Applications (VBA) macro. Beginning and intermediate SAS® programmers will benefit from learning an approach that was implemented to create basic Excel workbooks for each of forty research sites in a Veterans Affairs clinical research trial. While there are many techniques that can be used in SAS®, the example offered in this paper utilizes PROC REPORT, ODS ExcelXP Tagset, the SAS® Macro Facility and a VBA macro to convert Excel XML files to standard Excel workbooks. Although this example involves the reporting of non-adherent study participants in a large clinical research trial, the technique and associated SAS® code can be adapted for SAS® programmers in a variety of environments who need to automate reporting for multiple entities.

INTRODUCTION

When faced with the challenge of needing to create separate report files for each site in a nationwide clinical research study, a programmer is grateful for tools that can easily loop through data and make quick work of the task. In the CONFIRM study which currently has over 25,000 Veteran volunteers at forty sites across the United States and Puerto Rico, it is important for research personnel to be informed when study participants are not adhering to the clinical interventions of the study. In our study this means a participant agrees to either go have a colonoscopy done or send in a lab sample for testing depending on their randomization. In order to protect the internal validity of the study and ensure participants receive the best clinical care possible while enrolled, reports are distributed to each research site on a regular basis. Research staff can then intervene by using the Department of Veterans Affairs (VA) electronic health record system to prompt physician awareness and alter clinical care as needed. Using SAS® as the primary data collection and management system, programs are run routinely to track participants' progress in the study and to identify if a participant has not completed a recommended clinical procedure within a desired timeframe.

This paper demonstrates a simple way of generating and separating out a basic report file for any research site where one or more non-adherent participants have been identified. For the example that follows, a input SAS® data set is named PharmaSUG2015 and contains one record per participant. Each of the famous Veterans in PharmSUG2015, symbolic of the actual research study participants are assumed to have been identified beforehand with SAS® as being non-adherent to the study protocol. Each record contains the participant's unique study ID, Siteno (two digit numeric) to which they belong, first and last names, LastFour of their social security number and date of randomization (DateRand).

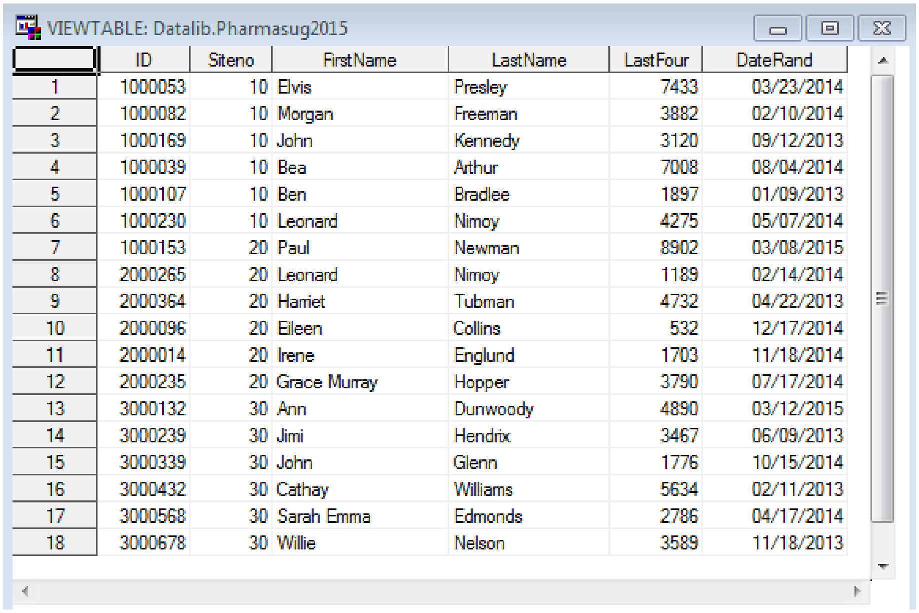
The goal is to create a simple Excel report file that research coordinators at each study site can use, as a tool to increase protocol adherence thereby ensuring the investment in the randomized control will yield valid, applicable results. While the report file produced in this example has been simplified for this paper, the principle and technique used to create it can be used for reports as complex as your needs require. The method employed involves the following Steps:

1. PROC SQL is used to determine the sites needing a report.
2. The macro SiteReport is called one time for each site with a non-adherent participant and utilizes the REPORT procedure in combination with the ODS ExcelXP tagset to create multiple XML report files that can be opened in Excel.
3. A Visual Basic macro runs in Excel totally independent of SAS®, and converts the Excel readable XML report files to files with the standard .xlsx extension instead of .xml.

STEP 1: DETERMINE WHICH RESEARCH SITES NEED A REPORT

Each VA research site is assigned a distinct two-digit numeric identifier named Siteno. Siteno serves as the classification variable within an if statement to select participants from each unique research site. Any of the sites that have at least one non-adherent participant will result in an Excel report being generated by the macro SiteReport. Depending on your application, any grouping variable could be substituted and used to drive a similar SAS® macro.

To keep things simple for this example, the input file PharmaSUG2015 seen in Display 1 contains eighteen participants from three distinct sites.



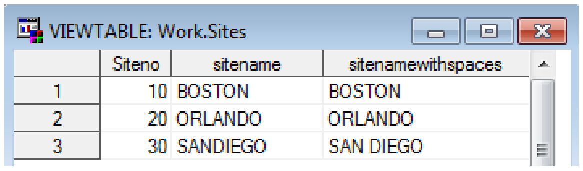
VIEWTABLE: Datalib.Pharmasug2015

	ID	Siteno	FirstName	LastName	LastFour	DateRand
1	1000053	10	Elvis	Presley	7433	03/23/2014
2	1000082	10	Morgan	Freeman	3882	02/10/2014
3	1000169	10	John	Kennedy	3120	09/12/2013
4	1000039	10	Bea	Arthur	7008	08/04/2014
5	1000107	10	Ben	Bradlee	1897	01/09/2013
6	1000230	10	Leonard	Nimoy	4275	05/07/2014
7	1000153	20	Paul	Newman	8902	03/08/2015
8	2000265	20	Leonard	Nimoy	1189	02/14/2014
9	2000364	20	Harriet	Tubman	4732	04/22/2013
10	2000096	20	Eileen	Collins	532	12/17/2014
11	2000014	20	Irene	Englund	1703	11/18/2014
12	2000235	20	Grace Murray	Hopper	3790	07/17/2014
13	3000132	30	Ann	Dunwoody	4890	03/12/2015
14	3000239	30	Jimi	Hendrix	3467	06/09/2013
15	3000339	30	John	Glenn	1776	10/15/2014
16	3000432	30	Cathay	Williams	5634	02/11/2013
17	3000568	30	Sarah Emma	Edmonds	2786	04/17/2014
18	3000678	30	Willie	Nelson	3589	11/18/2013

Display 1. The input data set PharmSUG2015.

From this input data set, the SQL procedure can be used to create the data set Sites which is shown in Display 2. Using the site format, variables sitename and sitenamewithspaces are created to be used within the SiteReport macro. The SAS® code for STEP 1 is as follows:

```
data PharmaSUG2015 ;
set PharmaSUG2015 ;
format FullName $36. ;
FullName = trim(LastName) || ', ' || trim(FirstName); *Concatenate first and last;
run ;

Proc format ;
value site    10 = 'Boston'
              20 = 'Orlando'
              30 = 'San Diego';
run ;

PROC SQL;
    create table Sites as
    select distinct site
    from work.PharmaSUG2015
    upcase(compress(put(siteno,site.),'. ')) as sitename,
    upcase(trim(put(siteno,site.))) as sitenamewithspaces
    order by site;
quit;
```



VIEWTABLE: Work.Sites

	Siteno	sitename	sitenamewithspaces
1	10	BOSTON	BOSTON
2	20	ORLANDO	ORLANDO
3	30	SANDIEGO	SAN DIEGO

Display 2. Sites becomes input for the _NULL_ data step in STEP 2.

STEP 2: EXECUTE SAS® MACRO TO CREATE EXCEL REPORT FOR EACH SITE

In order to run the SiteReport macro both the PharmaSUG2015 data set and the Sites data set are needed as inputs. Sites is processed a record at a time with a _NULL_data step. The CALL SYMPUT routine creates macro variables siteno, sitename, and sitenamewithspaces for usage in SiteReport. Following the creation of these variables, SiteReport is invoked by the CALL EXECUTE routine. The SAS® code which runs the macro for each siteno is as follows:

```
data _null_;
set Sites;
  call symput ('siteno',siteno);
  call symput ('sitename',trim(sitename));
  call symput ('sitenamewithspaces',trim(sitenamewithspaces));
  call execute('%SiteReport');   /* Run macro SiteReport for each site */

run;
```

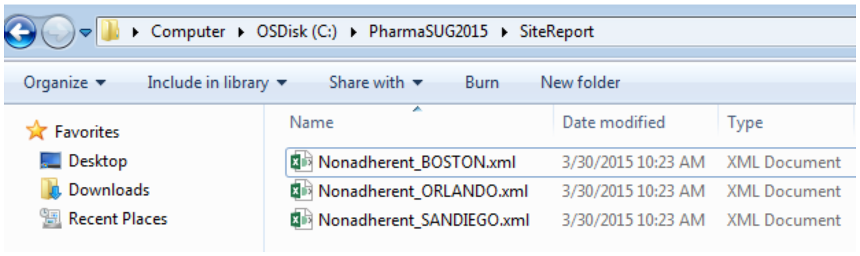
The SAS® macro SiteReport is straightforward and incorporates basic ODS tagsets ExcelXP coding in combination with PROC REPORT to produce an Excel readable output file in XML format. The ExcelXP tagset was chosen because our research personnel are most comfortable working with Excel files. In addition to being easy to print out, the spreadsheet column format gives the user the ability to add their own columns for tracking or auditing purposes (e.g. comments or action dates). This paper does not go into the details of the ODS ExcelXP but for particularly excellent papers on the topic, Vince Del Gobbo has been writing and presenting on this topic for several years and references have been provided. His papers are especially thorough at explaining many of the options available. The tagset options used in the example below include sheet_name which names the sheet in Excel, embedded_titles which ensures the SAS® titles are displayed in the worksheet instead of as an Excel print document header, autofit_height which makes the Excel row height adjust to the contents inside, absolute_column_width which gives the programmer the ability to control the column width sizes in Excel, and width_fudge which based on trial and error seemed to yield benefits despite explicitly naming specific column widths. The default width_fudge factor is 0.75 inches so 0.50 results in slightly narrower columns. All of these options remain in effect until they are explicitly changed or are cancelled by closing the ODS tagset ExcelXP destination.

```
%macro SiteReport ;

data work.Nonadherent_&sitename;
Set  work.PharmaSUG2015;
if   siteno = &siteno ;
run ;

ods tagsets.ExcelXP file = "C:\PharmSUG2015\SiteReport\Nonadherent_&sitename..xml"
                            style=CustomPrint;
ods tagsets.ExcelXP options(sheet_name = 'Nonadherent Participant Report'
                            embedded_titles='Yes'  autofit_height='Yes'
                            absolute_column_width='11,22,7,11'
                            width_fudge='0.50') ;

title1 justify=left "PharmaSUG Nonadherence Listing Report" ;
title2 justify=left " " ;
title3 justify=left "Study Site: &sitename   Run Date: %sysfunc(date(),mmddyy10.)";

proc report data = work.Nonadherent_&sitename nowd split='*' ;
column Id FullName LastFour DateRand;      /* Columns to be output in report*/

define Id / display center style(column)=[just=center] 'Participant*ID';
define FullName / display left 'Participant Name*(Last, First)';
define LastFour / display left 'Last*Four'
                 style(column)=[tagattr='format:0000'] /* Keep leading zeros*/
define DateRand / display center 'Randomized*Date' format=mmddyy10. ;
run;

ods tagsets.ExcelXP close;   /* Terminate ODS ExcelXP destination */

%mend SiteReport;
```

The PROC REPORT step within the macro simply displays four columns of data but could be made as complex as needed. Additionally the macro does not have any parameters, but lends itself to having a parameter for the file name, a parameter for the folder path and whatever else one chooses to make it more flexible. Note that this example uses a small modification of the Printer style to change the font to and make the background color white. The PROC TEMPLATE code that was used to make this modification is as follows:

```
proc template ;
edit styles.printer as CustomPrint ;
   style fonts /
       'TitleFont' = ("TimesRoman",12pt,Bold) ;
    style colors /
       'headerbg' = color_list('bg') ;
end; run;
```

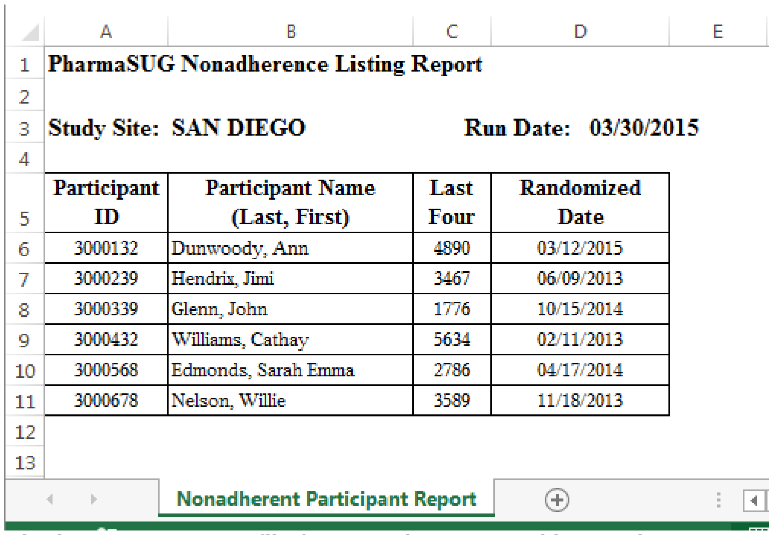
Each of the three output files generated by the ODS Tagset ExcelXP are written to a folder named SiteReport as specified by the file statement and can be seen below in Display 3.



Name	Date modified	Type
Nonadherent_BOSTON.xml	3/30/2015 10:23 AM	XML Document
Nonadherent_ORLANDO.xml	3/30/2015 10:23 AM	XML Document
Nonadherent_SANDIEGO.xml	3/30/2015 10:23 AM	XML Document

Display 3. XML files created after calling SiteReport macro for each site.

Double click on any of these basic report files and Excel will open normally as if it was a standard formatted workbook. As an example the XML report for our fictitious participants recruited at the San Diego research site has been opened and is shown in Display 4.



PharmaSUG Nonadherence Listing Report

Study Site: SAN DIEGO   Run Date: 03/30/2015

Participant ID	Participant Name (Last, First)	Last Four	Randomized Date
3000132	Dunwoody, Ann	4890	03/12/2015
3000239	Hendrix, Jimi	3467	06/09/2013
3000339	Glenn, John	1776	10/15/2014
3000432	Williams, Cathay	5634	02/11/2013
3000568	Edmonds, Sarah Emma	2786	04/17/2014
3000678	Nelson, Willie	3589	11/18/2013

Display 4. XML report file for San Diego opened in Excel.

Note that cells within the report have a solid black border but this attribute could be changed depending on personal preference. For more information on ways to change formatting in your Excel report an excellent paper by Asrani is noted in the references.

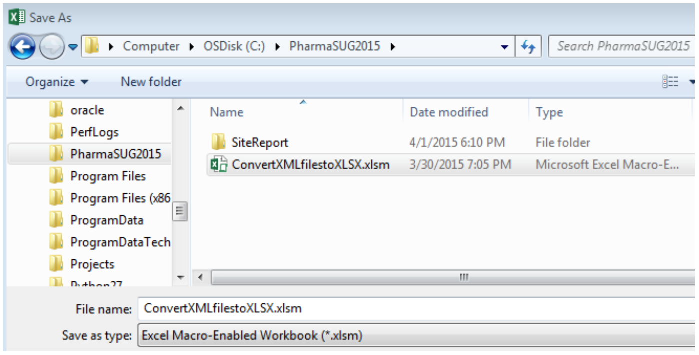
STEP 3: CREATE A VISUAL BASIC MACRO TO CONVERT ALL XML FILES IN A FOLDER TO STANDARD XLSX FORMAT FILES

As Display 4 demonstrates, the XML files that are created by the ODS ExcelXP tagset in step two can be opened by Excel in the XML format and display the same as any standard workbook with an XLSX file extension. This may suit your purposes and if this is the case the third step to create a Visual Basic macro is optional. However if you will be sharing the files with users, my recommendation is that the files be saved to the standard Excel (2007 or later) file format with the XLSX extension before distributing the files. Although in our study we post the output files to a VA intranet Sharepoint site, Springer (2011) points out, the XLSX format may work better with some email programs that do not work as well with XML files attached. Additionally more experienced users may notice the file extension distinction and prefer the usual file format. An easy way of changing the format of the XML file in Display 4 to XLSX is to open the XML file in Excel and then select **Save As** from the **File** menu and then select the XLSX format. However when there are many files to change this extra step can become very tedious to do multiple times. Having familiarity with Visual Basic this writer opted to create a macro in a Excel worksheet which can convert a batch of XML files with a single click. Step 3 explains the technique needed to implement a macro in an Excel file that can be reused over and over again.

There are different ways to accomplish this task just like there are usually many ways of accomplishing a programming goal in SAS®. As it turns out one of the alternative methods is a technique the author stumbled upon while writing and researching this paper. The approach, not described in this paper, allows a programmer to remain in the SAS® environment by executing a SAS® macro that uses VBScript (a subset of Visual Basic that can be executed in a different host environment). To learn more about this method, the paper written by Chevell Parker is provided in the references and his macro is available for download from the SAS Knowledge Base page here. If you are willing to move out of your SAS® comfort zone, the remainder of the paper offers directions for creating a Visual Basic macro named ConvertXMLtoXLSX that can be further customized to your requirements.

OPEN A NEW EXCEL SPREADSHEET AND ADD A SHAPE

Using Microsoft Excel version 2007 or later, open a new blank workbook. This workbook will be used to save your macro to be used as many times as you would like in the future for regular processing of new report files. To save the workbook, click **File**, click **Save As**, browse to the location you would like to save the file, type the file name and before saving change the file type from the default XLSX extension to the XLSM since the workbook will contain a macro. Notice in Display 5, the workbook is given the name ConvertXMLtoXLSX.xlsm.



Name	Date modified	Type
SiteReport	4/1/2015 6:10 PM	File folder
ConvertXMLfilestoXLSX.xlsm	3/30/2015 7:05 PM	Microsoft Excel Macro-E...

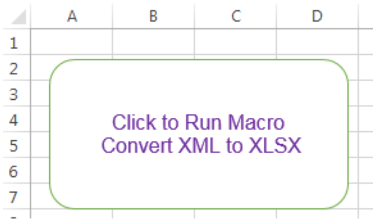
File name: ConvertXMLfilestoXLSX.xlsm
Save as type: Excel Macro-Enabled Workbook (*.xlsm)

Display 5. Excel macro-enabled workbook which will contain the Visual Basic macro.

Note you may need to enable the macro content depending on your security settings in Excel. To prevent any security warnings for your new workbook (steps here are for Excel 2013), click **File**, click **Options**, click **Trust Center**, click the **Trust Center Settings** button, click **Macro Settings**, and change the radio button to **Enable All Macros**.

You are now ready to add a shape or button to the worksheet, and this shape will be used to run the macro. The idea is that you will open the workbook and then just need to click on the shape to launch the macro. To add a shape in Excel the steps are the following :

- On the **Insert** tab, in the **Illustrations** group, click **Shapes**.
- Click the shape you like (rectangle with rounded corners chosen for this example), click anywhere in the workbook, and then drag to place the shape.
- Right click the shape and click **Add Text**.
- Type your desired text, then click outside the shape. Your text becomes part of the shape as in Display 6.

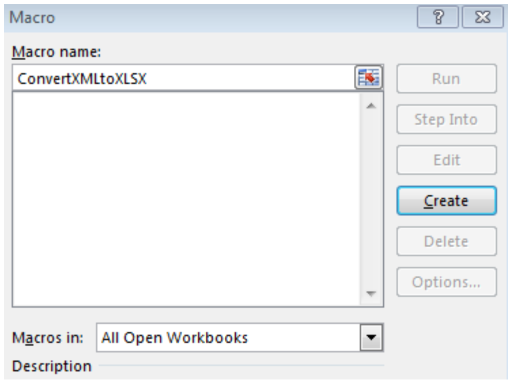


Display 6. Shape added to workbook which will be clicked to run Visual Basic macro.

ADD THE VISUAL BASIC MACRO TO THE EXCEL WORKBOOK

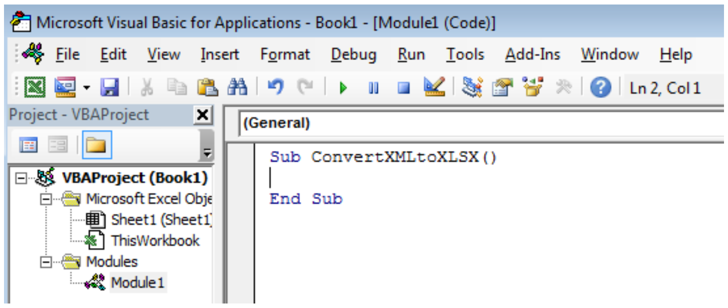
To create a new macro you will need to click on the Developer tab which may or not be present on your ribbon. By default the Developer tab does not show on the ribbon but it is very easy to make it show for usage with macros. If you are using Excel 2010 and 2013 click File, click Options, click Customize the Ribbon, select the Main Tabs from Customize The Ribbon drop down box, and check the **Developer** item. The technique for Excel 2007 is similar and can be found here.

Once the Developer tab is available, click **Macros**, and the window shown in Display 7 will appear. Type the name of your macro in the **Macro name** box, which for this example is named ConvertXMLtoXLSX.



Display 7. Window in Excel for naming and creating macro ConvertXMLtoXLSX

Click the create button and the Microsoft Visual Basic for Applications (VBA) window as seen in Display 8 will open and is ready for you to enter your Visual Basic code.


Microsoft Visual Basic for Applications - Book1 - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 2, Col 1
Project - VBAProject
(General)
VBAProject (Book1)
Microsoft Excel Obje
Sheet1 (Sheet1
ThisWorkbook
Modules
Module1

```
Sub ConvertXMLtoXLSX()

End Sub
```

Display 8. VBA window used to enter code for macro ConvertXMLtoXLSX

The entire macro is available in this paper and can copied into the VBA window. Delete the Sub and End Sub lines that Excel already inserted and copy and paste the Visual Basic code which follows:

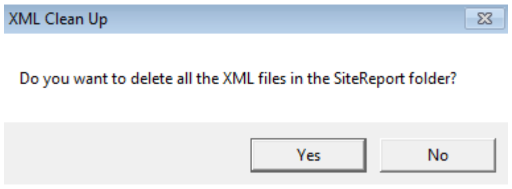
```
Sub ConvertXMLtoXLSX()
    Dim Filename, Pathname, saveFileName As String
    Dim wb As Workbook
    Dim i As Integer
    Dim iRet As Integer
    Dim strPrompt As String
    Dim strTitle As String
    Pathname = "C:\PharmaSUG2015\SiteReport\"
    Filename = Dir(Pathname & "*.xml")

    i = 0
    Do While Filename <> ""
        Set wb = Workbooks.Open(Pathname & Filename)
        saveFileName = Replace(Filename, ".xml", ".xlsx")
        wb.SaveAs Filename:=Pathname & saveFileName, _
                  FileFormat:=xlOpenXMLWorkbook, CreateBackup:=False
        wb.Close SaveChanges:=False
```

```
            Filename = Dir()
            i = i + 1
        Loop

        If i > 0 Then
             strPrompt = "Do you want to delete all the XML files in SiteReport?"
               'Dialog Title
             strTitle = "XML Clean Up"
               'Display MessageBox
            iRet = MsgBox(strPrompt, vbYesNo, strTitle)
               'Check pressed button
            If iRet = vbNo Then
               'Do Nothing
            Else
               'Careful: Following line deletes ALL xml files in the SiteReport folder
               Kill "C:\PharmaSUG2015\SiteReport\*.xml"
               MsgBox "You have successfully converted " & i & " XML files to XLSX ", _
               "files", vbInformation
            End If
        Else
            MsgBox "There are no XML files in the SiteReport folder to convert to ", _
               "XLSX", vbInformation
        End If
    End Sub
```

Due to space limitations explain the Visual Basic code for this macro in detail is not possible, but hopefully alterations such as the location of your XML files (in this example C:\PharmaSUG2015\SiteReport) can be made simply given the readability of the Visual Basic language. To save the macro just close the VBA window by clicking on the red 'X' in the top right hand corner of the window. When you resave the workbook, the macro will be saved with it as well.

ASSIGN THE MACRO AND RUN IT

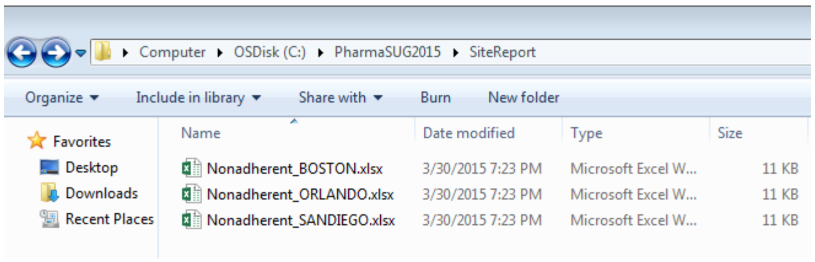
In order to assign the macro to the shape (Display 6) we created above, right-click on your shape and select **Assign macro**. A window similar to the one in Display 7 will appear. Choose the macro in the list and click **OK**. Now by simply clicking on the shape the macro ConvertXMLtoXLXS will run. It first creates each of the XLSX files in the SiteReport folder (macro could be changed to indicate a different destination folder) and then asks the user if the XML files should be deleted as can be seen from Display 9.


XML Clean Up

Do you want to delete all the XML files in the SiteReport folder?

Display 9. Message box to determine if XML files should be deleted.

For our example if we Click Yes, the three report files shown in Display 3 will be deleted and the three XLXS files will remain as can be seen in Display 10. Open any of the workbooks and the report will look the same as it did when the XML version of the file was opened (Display 4).



Name	Date modified	Type	Size
Nonadherent_BOSTON.xlsx	3/30/2015 7:23 PM	Microsoft Excel W...	11 KB
Nonadherent_ORLANDO.xlsx	3/30/2015 7:23 PM	Microsoft Excel W...	11 KB
Nonadherent_SANDIEGO.xlsx	3/30/2015 7:23 PM	Microsoft Excel W...	11 KB

Display 10. XLXS files that have been created by macro ConvertXMLtoXLSX.

The files are now ready to be distributed or posted to your intranet. As a final step, save and close the workbook containing the newly created macro ConvertXMLtoXLSX so it can be used the next time it is needed.

CONCLUSION

SAS® offers various ways of solving problems and it is up to the programmer to look for the most efficient solutions. For this programmer, finally having a real world need to use ODS Tagsets ExcelXP presented an opportunity to explore a tool that was long overdue to try. If you have been reluctant to try it as well perhaps this paper will give you some ideas on ways to make your work a little easier. Sometimes just the tools provided by SAS® will get the job done, but in other cases SAS® can lead us to experiment with other complementary tools like the Visual Basic macro offered in this paper. If Visual Basic is not a current part of your programming arsenal, you will likely find that by becoming familiar with it, there will be other future opportunities to automate Microsoft Office files.

REFERENCES

Asrani, Deepak. *SAS® Output Delivery System ExcelXP Tagset: Customizing Cell Patterns, Borders, and Indention.* Paper 095-2012, SAS® Global Forum 2012. http://support.sas.com/resources/papers/proceedings12/095-2012.pdf

Del Gobbo, Vincent. (2009) *More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®.* http://support.sas.com/rnd/papers/index.html#excel2009

Parker, Chevell. *The Perfect Marriage: The SAS® Output Delivery System (ODS) and Microsoft Office.* Paper SAS-TT02, PharmaSUG 2011. http://www.pharmasug.org/proceedings/2011/SAS/PharmaSUG-2011-SAS-TT02.pdf?page=12

Springer, Gayle. *Zebras Wanted: Using ODS Tagsets and PROC REPORT to Create a Striped Spreadsheet.* Coders' Corner, NESUG 2011. http://www.lexjansen.com/nesug/nesug11/cc/cc33.pdf

ACKNOWLEDGMENTS

The author would like to thank the Veterans who have volunteered for the CONFIRM study, the CONFIRM principal investigators, Drs. Robertson and Dominitz, all of the dedicated CONFIRM research team, and my talented programming colleagues who are always willing to make time for questions and to offer technical assistance and support. Thanks especially to Peter Guarino, Gary Johnson and Peggy Antonelli for their personal and professional support since my affiliation with the West Haven Cooperative Studies Program began eight years ago. Their backing of our attendance at various SAS® conferences has been invaluable. Finally thanks also to all the SAS® authors who have shared their work and made my job a lot easier over the years.

CONTACT INFORMATION

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

John O'Leary
Veterans Affairs Connecticut Healthcare
950 Campbell Avenue, MZ 151B
West Haven, CT 06419
203-932-5711 ext. 2402
john.oleary@va.gov