Paper 276-2012

Multi-sheet Workbooks using the ODS ExcelXP tagset

Cynthia A. Stetz, Bank of America Merrill Lynch, Hopewell NJ

ABSTRACT

Most of us are engaged in providing data to information consumers at least some of the time, and by far the most often requested format is the Microsoft Excel workbook. As the capabilities of Excel have expanded, so have the requests for more and more sophisticated output, and the search for ways to generate this output with the least amount of human effort as possible.

One of the SAS® tools that I have recently started using to great effect is the ODS ExcelXP tagset. By harnessing some of the vast capabilities available within this tagset, in concert with the judicious application of macro code and Dynamic Data Exchange (DDE), I am now able to deliver nicely formatted, multisheet native Excel workbooks for any number of subsets of my data as might be desired.

INTRODUCTION

This paper will explore how to use some of the ExcelXP tagset options to allow the REPORT procedure to produce XML files containing multiple sheets within individual workbooks. In addition, it will illustrate how to have a subsequent DDE step open and save the XML into native XLS format, which allows for the dynamic application of a password to the file. Finally, by wrapping all of this code in a macro loop, we can produce multiple output files, all with minimal manual effort.

BACKGROUND

We had already been using SAS to create reports, using the EXPORT facility to write out simple XLS files containing all data for one report in one sheet. We could create multiple workbooks using a macro loop, but did not have a simple way in SAS to create multiple sheets within a workbook. To accomplish that, which was the customer's requirement, a colleague of mine had written an Excel macro. The macro opened an Excel template file that contained all of the column formatting needed, imported the XLS file, split the one sheet into multiple sheets, applied a password, and then wrote the formatted file out again. It was a very robust macro and accomplished the goals for the customer. The only limitation was that it took quite a while to run, between 1 and 2 hours depending on how much data was involved, and while it was running, you could not use Excel or it might cause the process to hang. At that point you could either opt to start over, or manually edit some of the files the macro used, and re-start it from after the last file it processed successfully. My colleague had been running it primarily off-hours, in which case the limitations were easily managed. But our team expanded and so did the work load and we soon needed a new solution.

DISCOVERING THE EXCELXP TAGSET

Whenever I face a challenge, I head for the SAS website, type my question into the 'search' text box, and then read through the results till I find what I need. This turned up many excellent papers on ODS and the ExcelXP tagset, including several by Vincent DelGobbo, Eric Gebhart and Cynthia Zender. After much trial and error, I was able to write the code to give me what I needed.

Here is the code to produce multi-sheet Excel workbooks using the ODS ExcelXP Tagset

```
ods listing close;
ods tagsets.excelxp file="C:\Myfolder\myfilename.xml"
options(embedded_titles='yes'
       frozen_headers='5'
       orientation = 'Landscape'
```

Multi-sheet Workbooks using the ODS ExcelXP tagset, continued

```
        page_order_across='Yes'
        suppress_bylines='Yes'
        sheet_interval='bygroup'
        autofit_height='Yes'
        pages_fitwidth='1'
        pages_fitheight='9'
        FitToPage="yes"
        absolute_column_width="&COLS"
        sheet_label=' '
        pagebreaks='yes'
        doc='no'
        row_repeat='1-5'
        autofilter='ALL'
             )
   style=styles.sansprinter;
title1 "&distnm &campaigndesc";
title2 "&criteria" ;
title3 "Data as of &monend";
footnote1 'This document contains confidential information and is not intended
for further distribution.';
footnote2 'Do Not download, unless absolutely necessary.';
footnote3 'Do Not save anywhere but to a secure ML Network Drive.';
proc report data=oneoutput nowindows
       style(header) = {background=cx99ccff foreground=black font_size = 8pt
       font_weight = bold}
        style(column) = {font_size = 8pt};
 by pntname1;
 column
   pntname1 offname1 fcoff1 FCNAME1 fcno1 pth1
        pntname offname fcoff fcno FCNAME Client_Name acctnum2
        &keep
        &specialistn pth2;
 define pntname1     / order noprint ;
 define offname1     / order noprint  ;
 define fcoff1       / order noprint ;
 define fcname1      / order noprint ;
 define fcno1        / order noprint ;
 define pth1         / order noprint;
 define pntname      / display  'Complex' ;
 define offname      / display  'Office Name' ;
 define fcoff        / display  'Office Number' ;
 define fcno         / display  'FA Number'
               style(column) = {tagattr='format:@'};
 define fcname       / display  'FA Name';
 define Client_name  / display  'Client Name';
 define acctnum2     / display  'Account Number'
               style(column) = {tagattr='format:@'} ;
  /*** begin custom vars ***/
  %include "&campaignpath.\Sas Programs\_ProcReportCustomVars.sas";
  /*** end custom vars ***/
 define &specialistn / display 'Specialist';
 define pth2         / display 'PATH Core ID'
               style(column) = {tagattr='format:@'} ;
run;
ods tagsets.excelxp close;
ods listing;
run;
```

Let me mention at this point that there are basically two ways to create multi-sheet Excel workbooks using the ExcelXP tagset: you can open the ODS output file, run a procedure or DATA step to write out some data, issue another ODS sheet statement, and then run another procedure or data step. You can add as many sheets to the workbook as you like in this way. The final ODS close statement will end the workbook. Here is an example of this method using two PROC REPORT steps:

Multi-sheet Workbooks using the ODS ExcelXP tagset, continued

```
title1 ;
ods _all_ close;
ods tagsets.excelxp  file="&Pdrive\&campaign Privacy QC.xls"
style=styles.sansprinter;

ods tagsets.excelxp  options(sheet_label = 'Campaign Overview');

proc report missing data=Campoverview nowd
       style(header) = {background=cx99ccff foreground=black font_size = 8pt
       font_weight = bold}
       style(column) = {font_size = 8pt};

    column ("Campaign Overview" ordvar var1 var2);
    define ordvar / order noprint;
    define var1 / display '';
    define var2 / display '';
 run;

ods tagsets.excelxp options(sheet_label = "&campaign QC" );

proc report missing data=quality.QC_&campaign nowd
       style(header) = {background=cx99ccff foreground=black font_size = 8pt
       font_weight = bold}
       style(column) = {font_size = 8pt};

    column ("&campaign" "Privacy Scrubs"
            suppress pthcorid acctnum astycd
            hhaffiliateshare hhaffiliatemarketshare hhdnm hhdnm_bac);
       define suppress / display "Record Suppressed?";
       define pthcorid / display
                  style(column) = {tagattr='format:@'} "HH ID";
       define acctnum / display  "Account Number"
                  style(column) = {tagattr='format:@'};
       define astycd   / display "Account Source Type Code";
       define hhaffiliateshare / display "Affiliate Share";
       define hhaffiliatemarketshare / display "Affiliate Market";
       define hhdnm / display "ML DNM";
       define hhdnm_bac / display "BAC DNM";
run;

ods _all_ close;
ods listing;
```

The second way of producing multi-sheet Excel workbooks is to open the ODS ExcelXP tagset destination, set up some options to control the look of your worksheets, and then run a single procedure with a 'by' variable, which will become the values of your individual worksheets. This is the method I am using, and will explain more fully here.

OPEN THE ODS DESTINATION AND SET OPTIONS

Let's look at this part of the code:

```
ods listing close; [1]
ods tagsets.excelxp file="C:\Myfolder\myfilename.xml" [2]
    options(embedded_titles='yes'
             frozen_headers='5'
             orientation = 'Landscape'
             page_order_across='Yes'
             suppress_bylines='Yes'
             sheet_interval='bygroup'
             autofit_height='Yes'
             pages_fitwidth='1'
             pages_fitheight='9'
             FitToPage="yes"
             absolute_column_width="&COLS"
             sheet_label=' '
```

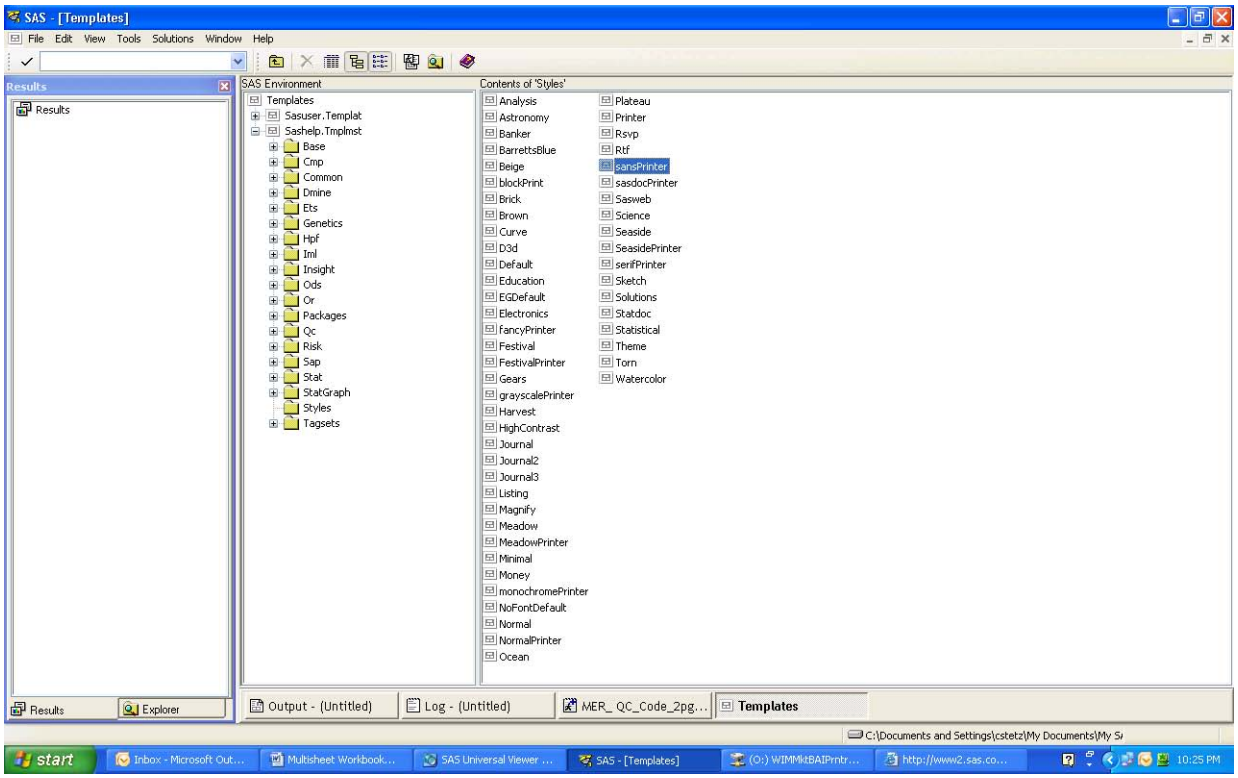
Multi-sheet Workbooks using the ODS ExcelXP tagset, continued

```
                pagebreaks='yes'
                doc='no'
                row_repeat='1-5'
                autofilter='ALL'
               ) [3]
style=styles.sansprinter; [4]
```

1. Close the default *listing* output destination to save resources
2. Open the *xml* output destination, specifying the filename and path.

A word about XML – 'XML (the e**X**tended **M**arkup **L**anguage) is an open standard for the definition, transmission, validation, and interpretation of data. The standard was developed by the Worldwide Web Consortium (W3C) in order to provide a simple and efficient way to manage self-documenting data files.' **(XML for SAS® Programmers** Frederick Pratter, Computer Science and Multimedia Studies Program, Eastern Oregon University, La Grande OR, SAS Global Forum 2008). And that's about all I'll say about that – learning the ins and outs of XML is beyond the scope of this paper, other than a mention I'll make at the end of how you can, if you are brave and adventurous enough, post-process your XML code to 'encourage' it to do some really nifty stuff. But I'm getting ahead of myself.

3. Specify all the options you need to control the look of your Excel workbook
4. Specify one of the pre-defined ODS styles that come with SAS, or, optionally, a customized one. I'm using one of the pre-defined SAS supplied styles here. You can see what styles are available by opening your Results window and navigating to Templates/SASHelp.Tmplmst/Styles. Here is a screenshot showing the sansprinter style hightlighted.



Display 1. ODS Template Styles available

XML OPTIONS

Multi-sheet Workbooks using the ODS ExcelXP tagset, continued

The XML options available in the ExcelXP tagset to customize your output are numerous. To see a complete list of all options written to your log, run your procedure with the option *doc='help'*. Since you will need to refer to this often, I recommend saving the log to a file. Now let's look at the ones I needed to get the output I wanted.

```
embedded_titles='yes'            Titles will appear in the worksheet, not as
                                 headers, which is the default.

frozen_headers='5                5 rows will be frozen as the sheet is
                                 scrolled.

orientation = 'Landscape'        Default is 'Portrait'.

page_order_across='Yes'          Default is 'down'.

suppress_bylines='Yes'           Bylines will not appear.

sheet_interval='bygroup'         A new sheet is created for each by group value

autofit_height='Yes'             Excel determines row height.

pages_fitwidth='1'               Fit to 1 page across.

pages_fitheight='9'              This is actually a 'fudge' - I want to allow a
sheet to be multiple pages long, but there is no tagset option to allow that.
So I set the value to '9', and then post-process the XML and re-set it to '0',
which is the XML value to accomplish what I want.

FitToPage="yes"                  This must be used with the pages_fit options.

absolute_column_width="&COLS"    Right now, PROC REPORT does not send the XML
enough information to properly set the column widths automatically, so we have
to do it explicitly.  I use a macro for flexibility.

sheet_label=' '                  The value of the BY group will be the sheet
                                 label


pagebreaks='yes'                 Page breaks will be inserted

doc='no'                         No need to list all the options every time we
                                 run this!

row_repeat='1-5                  Rows 1-5 will be repeated when a worksheet
                                 breaks across pages.

autofilter='ALL'                 Auto filter will be applied to all columns.
```

THE REST OF THE STORY

Most of the rest of the PROC REPORT procedure code is fairly standard stuff. Exceptions to this are the style elements on the PROC REPORT statement, and the tagattr portion of the style element you see on some of the define statements.

```
proc report data=oneoutput nowindows
     style(header) =
{background=cx99ccff foreground=black font_size = 8pt font_weight = bold}[1]
     style(column) = {font_size = 8pt}[2];
```

1. These options allow me to control the color of the background of my headers, the foreground, and the font size and weight. Notice the use of 'CX' as a prefix on the RGB color value – this was required to match a previously used color.

2. For the columns, I am only specifying the font size and allowing all other values to default.

Multi-sheet Workbooks using the ODS ExcelXP tagset, continued

```
define pth2    / display 'PATH Core ID'
                 style(column) = {tagattr='format:@'} ;
```

Sometimes Excel does things 'for' us that we would rather it did not do. One of these is the striping off of leading zeros. This is quite problematic when dealing with variables containing zipcode or perhaps account numbers. By using the {tagattr='format:@'} on our column definition we can control this 'feature' of Excel and get the output we want!

SO, HOW AGAIN DO WE GET THE MULTISHEETS?

The combination of the Excelxp tagset option "sheet_interval='bygroup'" and using "by pntname1" in the PROC REPORT code basically gives you one sheet per by group value. You will need many of the other options to get the output just the way you want it, depending on your requirements.

USING DYNAMIC DATA EXCHANGE TO TURN XML INTO 'NATIVE' XLS

The XML files that are created are quite large, so the quicker we convert them into 'native' XLS files and remove the XML files the better. The quickest way to do this is to use a DDE, or Dynamic Data Exchange step. This is a very fancy way of saying 'have SAS open the file, do whatever you want to do to it, and then save the file, possible in a different file format.' In this case we want to add a password, which we have saved in a macro variable at the top of our code, with all our other macro variables. Here's what the DDE code looks like.

```
/*** open the XML with DDE, save it as an XLS and add a password ***/
options noxsync noxwait;
filename sas2xl dde 'excel|system';
data _null_;
  length fid rc start stop time 8;
  fid=fopen('sas2xl','s');
  if (fid le 0) then do;
    rc=system('start excel');
    start=datetime();
    stop=start+10;
    do while (fid le 0);
      fid=fopen('sas2xl','s');
      time=datetime();
      if (time ge stop) then fid=1;
    end;
  end;
  rc=fclose(fid);
run;

%let frmpath=&campaignpath.\SharePoint_Raw\divn&divn.\;
%let frmname=&distnm._&gekkofileprefix._pgs.xml;

%let savepath=&campaignpath.\Sharepoint_Fmt\divn&divn.\;
%let savename=&distnm._&gekkofileprefix..xls;

data _null_;
  length ddecmd $ 200;
  file sas2xl;
  ddecmd='[open("'||"&frmpath"||'\'||"&frmname"||'")]';
  put ddecmd;
  ddecmd='[save.as("'||"&savepath"||'\'||"&savename"||'",1,"'|| "&PW" ||'")]';
  put ddecmd;
  ddecmd='[QUIT()]';
  put ddecmd;
run;
```

Pretty neat, right? The trickiest part of this code was getting the syntax around the PW macro variable correct. I started with a static value instead of the macro variable, and figured it out from there. By careful with your quotes!

WRAP IT UP, I'LL TAKE IT HOME!

Multi-sheet Workbooks using the ODS ExcelXP tagset, continued

The last piece of this project was to wrap it all up in a macro, so we could have SAS create multiple workbooks, each with multiple sheets in them. This is a very good example of the power of the MACRO language!

```
/*** RunXMLRpts ******************************************/
*** run the report for each entity;
%macro RunXMLRpts(keep,Specialistn,Test); ** test with numreps set to 1;
 %if &Test = T %then %let numreps = 1;
 %else %do;
  ** get number of entities for which we are doing a report;
  proc sql noprint;
   select count(distcd) into :numreps from replist;
  quit;
 %end;
 %put There are &numreps reports;
 %do i = 1 %to &numreps;
     * populate macro var to select report population and for filename;
     data _null_;
          set replist;
          if _n_ = &i then do;
              call symput('divn',left(put(slsrgncd,$char6.)));
              call symput('distcd',left(put(distcd,$char6.)));
           call symput('distnm',trim(left(put(distnm,$char40.))));
        end;
         run;
    /*** subset to one district (Region) ***/
    /*** we need two copies of some variables so we can use one to
         control the order and the other to be displayed ***/
    data oneoutput;
      set foroutput(where=(distcd = "&distcd"));
       pntname1 = pntname;
       offname1 = offname;
       fcoff1 = fcoff;
       fcno1 = fcno;
       fcname1 = FCNAME;
       pth1 = pthcorid;
       pth2 = pthcorid;
    run;

%put &divn &distnm &distcd ;

/*** delete the file if it exists ***/
%DeleteExistingFile
 (&campaignpath.\SharePoint_Raw\divn&divn.\&distnm._&gekkofileprefix..xml);

ods listing close;

/*** PROC REPORT CODE GOES HERE ****/

ods listing;
run;
/*** end of loop - executes once for each region (distcd) ***/
%end;

%put RunXMLRpts is done;

%mend RunXMLRpts;
/*** end RunXMLRpts ***************************************/
```

The SAS program determines how many reports to create, from the actual data it is using for the reports, and executes the loop containing the rest of the code once for each value. The value of the workbook-level variable is extracted from the data and pushed into a macro variable, which makes it available for use in both the filename and the title. We include macro put statements to the log, to let us check the log later, perhaps with another program, and report on the successful execution of this program!

CONCLUSION

The ODS ExcelXP tagset allows you to create multi-sheet Excel files, via the XML language. Style sheets, tagset options, and style elements all allow for extensive customizing of the output, even to controlling some 'features' that Excel imposes on us. Since the XML is a text file, it is also possible to post-process the XML to customize it even further – I added 'hard' page breaks, based on another value within my data, in addition to modifying the 'pages long' value. This is for the more brave (or foolhardy) programmers among us, however, as it is very easy to 'break' your XML code! But, for most applications, the options that are available should be sufficient, and certainly are a big leap forward.

REFERENCES

More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®
Vincent DelGobbo, SAS Institute Inc, Cary, NC
Paper 152-2009 SAS Global Forum 2009

ODS ExcelXP: Tag Attr Is It! Using and Understanding the TAGATTR= Style Attribute with the ExcelXP Tagset
Eric Gebhart, SAS Institute Inc., Cary, NC
Paper 031-2010 SAS Global Forum 2010

The Greatest Hits: ODS Essentials Every User Should Know
Cynthia Zender, SAS Institute Inc., Cary, NC, USA
Paper 300-2011 SAS Global Forum 2011

CONTACT INFORMATION

The author encourages your comments and questions and can be contacted at:

Cynthia A. Stetz
Bank of America Merrill Lynch
1700 Merrill Lynch Dr.
Hopewell, NJ
609-274-4461
Cynthia.stetz@bankofamerica.com or Cynthia_stetz@yahoo.com