A Simple Method of Creating Custom Excel Reports

Jimmy DeFoor
Citi Retail Bank

This paper will show the reader how to create formatted Excel-based reports without using SAS Styles. It will use the technique of creating a report-structured spreadsheet that imports data from a data-structured spreadsheet.

The SAS coding methods shown in this paper will include:

1) Using the Types statement in a Proc Summary to create line totals and report totals.
2) Creating user formats to provide descriptions for each report line, including minus ('-') signs that indicate the line should be subtracted from a prior line.
3) Building a dummy data set with every possible line in the report so that the report would always have a fixed number of columns and rows.
4) Establishing a macro that would create the same report structure for the three different summaries of the source data.
5) Exporting reports to different worksheets in the same spreadsheet using Proc Export with the sheet option.
6) Cleaning up existing spreadsheets and copying spreadsheets using the Call System option.
7) Send an email with Filename FTP announcing that the spreadsheet is available for review.

The Excel coding methods include:

1) Linking report cells to a data cells in another spreadsheet updated by the SAS program.
2) Bolding text in the Report Title and Column Cells.
3) Formatting linked report cells as needed.
4) Using the IF function and the ISBLANK function to prevent zeros in empty numeric cells.

Below are examples of the reports created in the report spreadsheet.

Report Example #1

Characteristics:

1) Formatted Title Column Headings and Title
2) Fixed number of rows and columns.
3) Data content, including month and year, is updated automatically.

Early Warning DM Population			**July 2018**
Category	**Accounts**		
Total DM Population	104,208		
- Charged Off	20,029		
- Closed	42,440		
- Missed Accts	178		
Early Warning DM Population	41,561		

Report Example #2

Characteristics:

1) Formatted Title Column Headings and Title
2) Fixed number of rows and columns.
3) Data content, including month and year, is updated automatically.

DM Tier 1 Early Warning Waterfall					**July 2018**
Review Group	**TIER**	**Accounts**	**Net Book**	**Commitment**	**Availability**
DM Early Warning Population	TIER1	37,391	409,874,179	931,939,529	1,350,875,201
- In Remedial	TIER1	0	0	0	0
- Charged Off	TIER1	0	0	0	0
- Closed Account	TIER1	3,385	3,881,095	23,477,429	19,659,188
- CALFED Closed Line Zero Bal	TIER1	738	0	530,000	20,000
- CALFED Closed Line Special	TIER1	391	2,635,357	0	0
- BCP Closed Line Any Bal	TIER1	4,592	1,226,773	2,928,577	18,313
- AFS Closed Line Zero Bal	TIER1	6	0	654,000	0
- AFS Closed Line Prod 21999	TIER1	0	0	0	0
- AFS Accts to be Closed	TIER1	1	0	10,000	0
- Past Due 6 days and more	TIER1	250	2,423,350	5,312,723	476,717
- In Forbearance	TIER1	0	0	0	0
- AFS Termed Out by Status	TIER1	3,101	99,911,304	102,911,329	1,424,351
- AFS Termed Out by Mat Date	TIER1	12	231,655	231,655	0
- Active Duty Military	TIER1	14	105,588	541,000	435,412
- Commercial Dispute	TIER1	0	0	0	0
- WEBCATS Exclusion	TIER1	148	5,873,913	10,277,361	4,434,792
- No Adverse Action Codes	TIER1	0	0	0	0
- Two Pct Min Pay - CALFED	TIER1	0	0	0	0
- Disaster	TIER1	0	0	0	0
Total Excluded Accounts	**TIER1**	**12,638**	**116,289,035**	**26,468,773**	**146,874,074**
- Risky Accts Not Actionable	TIER1	51	1,833,575	155,000	0
- Risky Accts Loaded for Revie	TIER1	119	5,664,019	6,114,536	1,926,060
- Currently Not Risky Accts	TIER1	24,583	286,087,550	1,197,731,591	903,544,696
Total Reviewed Population	**TIER1**	**24,753**	**293,585,144**	**905,470,756**	**1,204,001,127**

Note: DM means Decision Managed.

Report Example #3

Characteristics:

1) Formatted Title Column Headings and Title
2) Varying number of columns and rows (within bounds).
3) Data content, including month and year, is updated automatically.

Accounts Missing from DM Population					**July 2018**		
Obl_no	**Prod_Cd**	**Prod_Cd_Desc**	**Proc_tp**	**Book_Dt**	**Net_Book_Bal**	**Availability**	**Commitment**
954752	1807	BCP	9999	10/23/2014	0	0	0
1050755	1807	BCP	9999	7/9/2001	0	0	0
1724537	1807	BCP	9999	5/20/2005	0	0	0
1778658	1807	BCP	9999	7/1/1992	0	0	0
2955994	1807	BCP	9999	11/29/2006	0	0	0
3071701	1807	BCP	9999	3/26/2012	0	0	0
3249602	1807	BCP	9999	4/20/2012	0	0	0
4403165	1807	BCP	9999	5/11/2009	0	0	0
5051667	1807	BCP	9999	7/2/1999	0	0	0
5292292	1807	BCP	9999	12/20/2006	0	0	0
13495516	1807	BCP	9999	9/19/2006	0	0	0
13542087	1807	BCP	9999	4/12/2007	0	0	0
15454592	1807	BCP	9999	3/5/2009	0	0	0

Excel methods described by report.

Report Example #1

Early Warning DM Population	
Category	**Accounts**
Total DM Population	104,208
- Charged Off	20,029

Titles and column headings bolded in Report worksheet.

Data cells linked to related cells in spreadsheet created by SAS program.
(='I:\Documents\1_waterfall\[bws_waterfall_data.xlsx]STARTING_POP'!B2)

Month and Year on each report.

July 2018

Report month linked to its own worksheet.
(='I:\Documents\1_waterfall\[bws_waterfall_data.xlsx]Report_Month'!A2

The same methods were used in Report Example #2.

Report Example #3

Lists every account not found in the DM population.

Accounts Missing from DM Population					July 2018		
Obl_no	**Prod_Cd**	**Prod_Cd_Desc**	**Proc_tp**	**Book_Dt**	**Net_Book_Bal**	**Availability**	**Commitment**
954752	1807	BCP	9999	10/23/2014	0	0	0
1050755	1807	BCP	9999	7/9/2001	0	0	0
1724537	1807	BCP	9999	5/20/2005	0	0	0
1778658	1807	BCP	9999	7/1/1992	0	0	0

Excel methods used

1. Bolded fixed titles and column headings in Report worksheet.
2. Used IF function and ISBLANK function to prevent zeros in empty numeric cells.
   IF(ISBLANK('I:\Documents\1_waterfall\[bws_waterfall_data.xlsx]MISSING_POP'!A2)," ",
   'I:\Documents\1_waterfall\[bws_waterfall_data.xlsx]MISSING_POP'!A2))

Every possible cell that can be populated, or every cell of a good sample of the missing records, is populated with the IF function containing the ISBLANK function on the Accounts Missing report.

General method of creating the Excel report worksheets.

1) Open the data spreadsheet and report-to-be spreadsheet at the same time.
2) Create your titles and column headings in each worksheet. Bold and font as desired.
3) Find which cells will contain your report data.
4) Start at the left top cell of your report area and enter an = sign. Then move your cursor to the data cell that you wish to import and click that cell to activate it. Tap the Enter key after that. You should now see the data in the report cell that you linked. The formula bar will also show the link reference, such as '='I:\Documents\1_waterfall\[bws_waterfall_data.xlsx]STARTING_POP'!B2'.
5) If the cell has the $ in front of each element of the linked cell reference, such as B2, remove the $.
6) Copy the full link reference to each cell of the report region.
7) Format each column as appropriate for its contents.
8) If the source cell can be blank and the report cell is numeric, then you will get zeros in your linked cell when the source cell is blank. To avoid that, use the IF function with the ISBLANK function as described above.
9) After all links have been established and the formatting is complete, save the report spreadsheet in the same directory as the data spreadsheet.

Updating the Excel report spreadsheet from the data spreadsheet

1) Open the report spreadsheet
2) Under the Data tab, open the Edit Links option.
3) Select 'Check Status'. The status will become 'OK'. Next, select 'Update Values'.
4) If the data values are not updated, or a message is displayed that all of the links cannot be updated, then select 'Open Source'. The data spreadsheet will open and the values in the report spreadsheet will update. Close the data spreadsheet then.
5) After the report spreadsheet is updated, save it under another name, perhaps with the date as part of the name. Then, select 'Break Links' and save the data set again. You now have a report data set that represents the date that it was updated.

Next shown is the SAS code used in creating the data-structured spreadsheet that is linked to the report spreadsheet.

Create report date for each report

This code creates the data set with the report month and year, such as June 2018, that is used on every report. The data set will be exported to its own worksheet. A macro variable is also created with a Call Symput.

```
data report_month;
  date  = put(today(),worddate.0);
  month = strip(scan(date,1));
  year  = strip(scan(date,-1));
  month_year = strip(month)||' '||strip(year);
  keep month_year;
  call symput('rpt_mth',strip(month_year));
run;
```

Summarize data for Report Example #1.

Proc Summary summarizes by the source data by category and also creates the total line 'Total DM Population' in the Report Example. The '( )' in the Types statement creates the grand total line. It will have a _type_ number of zero (0) in the output data set 'summary_catgry'.

```
proc summary data=dm_population missing;
  class category;
  types () category;
  output out=summary_catgry;
run;
```

Summary_catgry

Category	_TYPE_	Accts
A	0	104062
B	1	19996
C	1	42753
D	1	220
E	1	41093

The _Type_ of zero (0) is the total line.

Next, the code assigns the category of 'A' to the total (0) line and creates the category description for each summary line. A user form is used to put the minus (-) sign next to the description of each category that must subtracted from the Total DM population. The user format also enables the ordering of the summary values in the form needed for the report.

```
data summary_catgry;
  length Category   $1
         Catg_desc $27;
  Rename _freq_ = Accts;
  set summary_catgry;
  if _type_ = 0 then
    category = 'A';
  catg_desc = put(category,$catg.);
  output;
run;
```

$catg. is the user format.

Typically, a sort would be required to order the categories, but not here since the zero (0) line is listed first in the summary output.

This is the Proc Format that generated the $catg format. User formats are efficient, easily-understood, and portable methods of describing data. They are a good coding method to apply in all of your SAS programs.

```
proc format;
  value $catg
    'A' = 'Total DM Population'
    'B' = ' - Charged Off'
    'C' = ' - Closed'
    'D' = ' - Missed Accts'
    'E' = 'Early Warning DM Population'
    ;
run;
```

After the application of the $catg. user format, the summary_catgry dataset looks like this:

Category	Catg_desc	_TYPE_	Accts
A	Total DM Population	0	104062
B	- Charged Off	1	19996
C	- Closed	1	42753
D	- Missed Accts	1	220
E	Early Warning DM Population	1	41093

Export report data to the data spreadsheet.

All of the xlsx output will go to one directory. To save typing and to enable the easy changing of the output destination on every Proc Export, the destination is assigned to a macro variable.

```
%let tdir = /usr04/ap/analyst/jd74575/1_miscellaneous/output;
```

Multiple worksheets will be written to data spreadsheet, so to ensure that no worksheets contain old data, the existing data spreadsheet is deleted. This is done with the Call System function and the 'rm' UNIX command.

```
data _null_;
  call system("rm &tdir./bws_waterfall_data.xlsx");
run;
```

Next write the data to the worksheets in the spreadsheet.

First, the report month is exported to the report_month worksheet in the waterfall spreadsheet. Notice that sheet option is used to write to name the worksheet that is populated. The range option is not available in 'Proc Export dbms=xlsx', so the data cannot be written to a specific group of cells.

```
proc export data=report_month dbms=xlsx
            outfile="&tdir./bws_waterfall_data.xlsx";
   sheet = report_month;
run;
```

Next, export the category summary to the starting_pop worksheet.

```
proc export data=summary_catgry(drop=_type) dbms=xlsx
            outfile="&tdir./bws_waterfall_data.xlsx";
   sheet = Starting_Pop;
run;
```

Then, export the accounts in the missing population to the missing_pop worksheet.

```
proc export data=missing_population dbms=xlsx
            outfile="&tdir./bws_waterfall_data.xlsx";
   sheet = Missing_Pop;
run;
```

The replace option isn't necessary on any of these exports because no sheet is being overwritten. All are being created. Despite that, the spreadsheet will be backed up when the second sheet is created. The Call System function can be used to delete the backup .

```
data _null_;
  call system("rm &tdir./bws_waterfall_data.xlsx.bak");
run;
```

Create the other summary data sets for the report.

This Proc Format creates the user format that will provide the descriptions of the lines in the next three reports.

```
proc format;
    value $grp
      '00' = 'DM Early Warning Population'
      '01' = '- In Remedial'
      '02' = '- Charged Off'
      '03' = '- Closed Account'
      '04' = '- CALFED Closed Line Zero Bal'
      '05' = '- CALFED Closed Line Special '
      '06' = '- BCP Closed Line Any Bal'
      '07' = '- AFS Closed Line Zero Bal'
      '08' = '- AFS Closed Line Prod 21999'
      '09' = '- AFS Accts to be Closed'
      '10' = '- Past Due 6 days and more'
      '11' = '- In Forbearance'
      '12' = '- AFS Termed Out by Status'
      '13' = '- AFS Termed Out by Mat Date'
      '14' = '- Active Duty Military'
      '15' = '- Commercial Dispute'
      '16' = '- WEBCATS Exclusion'
      '17' = '- No Adverse Action Codes'
      '18' = '- Two Pct Min Pay - CALFED'
      '20' = '- Disaster'
      '21' = 'Total Excluded Accounts'
      '22' = '- Risky Accts Not Actionable'
      '23' = '- Risky Accts Loaded for Review'
      '24' = '- Currently Not Risky Accts'
      '25' = 'Total Reviewed Population'
     other = 'Not Identified';
 run;
```

Again, notice the dashes (-) used to indicate subtraction.

Now, the scored population is summarized into each of the three summaries to be reported: Tier 1, Tier 2, and Total. The Total summary is generated by adding a second Type to the summary that covers just the variable 'review_grp'. The lower-level summaries of Tier 1 and Tier 2 are created by the Type 'review_grp *tiergroup'.

```
proc summary data=scored_population missing;
  class review_grp tiergroup;
  types review_grp*tiergroup review_grp;
  var net com_bal availability commitment;
  output out=summary_review_grp sum=;
run;
```

Next is the output of the summary procedure. Notice the blanks in tiergroup when _TYPE_ = 2;

Summary_review_grp

review_grp	tiergroup	_TYPE_	_FREQ_	net	com_bal	availability	commitment
01		2	542	-9819834	511334	0	511334
03		2	3382	6329084	26091647	22823548	26091647
04		2	735	0	510000	0	510000
05		2	386	2498370	0	0	0
06		2	4647	1198549	3013577	13729	3013577
07		2	27	0	3107616	72510	3107616
09		2	13	0	4585000	2719810	4585000
10		2	573	1031226	7527627	724772	7527627
12		2	3269	119226143	128020406	6472616	128020406
13		2	44	6956544	4363060	37357	4363060
14		2	17	511662	1416000	904029	1416000
16		2	134	11871518	14035694	3944331	14035694
22		2	95	17584147	45000	0	45000
23		2	323	40824097	23108531	5310712	23108531
24		2	26906	658850892	1622725572	1186498295	1622725572
01	TIER2	3	542	-9819834	511334	0	511334
03	TIER1	3	3291	3279445	20659025	18343842	20659025
03	TIER2	3	91	3049639	5432622	4479706	5432622
04	TIER1	3	732	0	510000	0	510000
04	TIER2	3	3	0	0	0	0
05	TIER1	3	386	2498370	0	0	0
06	TIER1	3	4551	1132389	2858577	13729	2858577
06	TIER2	3	96	66161	155000	0	155000
07	TIER1	3	13	0	1263563	0	1263563
07	TIER2	3	14	0	1844053	72510	1844053

The number of _TYPE_ is determined by the position of the variables in the Class statement and the presence of the variables in the Types statement. For a thorough explanation of how to calculate the number for a particular combination of Type variables, see Frank Ferriola's paper 'What's Your _TYPE_? How to find the CLASS You Want in Proc Summary'. It's website address is http://www2.sas.com/proceedings/sugi27/p077-27.pdf.

For Class statements involving four or less variables, this example can be used that uses 2 raised to the power associated with the position of the variable in the Class statement: 2^0,2^1, 2^2, 2^3. The last variable in the Class statement is the zero position, next is one, then two, etc. Sum the values for the combination of variables used in the Types statement.

```
Class   Var3    Var2    Var1    Var0;
/*      2^3 (8)  2^2 (4)  2^1 (2)  2^0 (1)  */
```

A Types statement with only Var2 would have a _Type_ of 2^2 (4). A Types statement with only Var1 would have a _Type_ of 2^1 (2). A Types statement of Var2 and Var1 would have a _Type_ of 6 (4+2). A Types statement of all four variables would have a _Type_ of 15 (8+4+2+1).

The Class and Types statements shown earlier were:

```
  Class review_grp tiergroup;
  Types review_grp*tiergroup review_grp;
```

Thus, as shown in our earlier data listing, the _Type_ values of the summary data will be 3 (2+1) and 2. This lets us assign 'Total' to Tiergroup when the _Type_ = 2.

```
data summary_review_grp;
  length Review_Grp $02
         Tiergroup  $06
         ;
  rename Freq_ = Accounts;
  Drop _type_ commitment;
  set summary_review_grp;
  if _type_ = 2 then
    tiergroup = 'TOTAL';
  output;
run;
```

Creating a dummy data set for the reports

To ensure that the number of rows in each report is always 22, zero-value lines must be created for each review_grp in a dummy data set that will be merged with the actual data. Since the Review_grps are numeric in form, this is easy to do with a Do Loop and a put statement that creates a character numeric. Lines 19 and 21 are not output because they do not exist in the Review_grps data.

```
data review_grp_zeroes;
  do j = 1 to 24;
    Review_Grp = put(j,z2.0);
    Accounts   = 0;
    Net        = 0;
    Com_Bal    = 0;
    Availability = 0;
    Commitment   = 0;
    drop j;
    if j not in (19,21) then
      output;
  end;
run;
```

Establish a macro to create the three reports from the same code.

The reports have the same structure, so using a macro to retain the code prevents repeating the code three times in the program. Only the macro name and parameter need be repeated.

The report code in the macro has six steps:

1) Retrieve the appropriate summary data from summary_review_grps, either Tier1, Tier2, or Total.
2) Merge the dummy zero-value data set against the retrieved summary data.
3) Create the additional summary lines needed for each report.
4) Assign the correct description to each line from the user format $grp.;
5) Sort the data so that the summary lines are in the correct order.
6) Export the data to the appropriate spreadsheet.

```
%macro grp_tier(level=);
    %let tier=%upcase(&level);
    data review_grp_&level;
      length Review_Grp $02
             Review_Grp_Desc $30
             Tiergroup  $06
             ;
      merge review_grp_zeroes (in=z) end=eof
            summary_review_grp(in=s  where=(tiergroup="&tier"));
      by review_grp;
      keep tiergroup review_grp review_grp_desc
           accounts net availability com_bal;
      review_grp_desc = put(review_grp,$grp.);
      if z then
        do;
           Availability = round(availability,1);
           Net          = round(net,1);
           Com_Bal      = round(com_bal,1);
           output;
           if review_grp le '20' then
            do;
               line21_frq_total + accounts;
               line21_net_total + net;
               line21_avl_total + com_bal;
               line21_com_total + availability;
            end;
           if review_grp ge '22' and
              review_grp le '24' then
            do;
               line25_frq_total + accounts;
               line25_net_total + net;
               line25_avl_total + com_bal;
               line25_com_total + availability;
            end;
           if review_grp le '24' then
            do;
               line00_frq_total + accounts;
               line00_net_total + net;
               line00_avl_total + com_bal;
               line00_com_total + availability;
            end;
          end;
```

Level= is a keyword parameter to receive the summary level wanted for each report.

End=eof creates an end-of-data flag variable.

User format $grp assigns line descriptions.

This code creates the additional summary lines needed for report.

```
      if eof then
        do;
           Accounts = line21_frq_total;
           Net      = line21_net_total;
           Availability = line21_avl_total;
           Com_Bal      = line21_com_total;
           Review_Grp   = '21';
           Review_Grp_Desc = put(review_grp,$grp.);
           output;
           Accounts = line25_frq_total;
           Net      = line25_net_total;
           Availability = line25_avl_total;
           Com_Bal      = line25_com_total;
           Review_Grp   = '25';
           Review_Grp_Desc = put(review_grp,$grp.);
           output;
           Accounts = line00_frq_total;
           Net      = line00_net_total;
           Availability = line00_avl_total;
           Com_Bal      = line00_com_total;
           Review_Grp   = '00';
           Review_Grp_Desc = put(review_grp,$grp.);
           output;
        end;
    run;
    *;
    proc sort data = review_grp_&level;
      by review_grp;
    run;
    *;
    proc export data=review_grp_&level dbms=xlsx
                outfile="&tdir./bws_waterfall_data.xlsx";
                sheet = &level;
    run;

%mend grp_tier;
```

This eof variable is set to one (1) at the end of the dummy data set.

The additional summary lines are then output.

&level comes from the level= keyword parameter in macro execution.

Next, the macro is used three times to generate the three waterfall summaries. Each time, a different level parameter is used.

```
%grp_tier(level=tier1);

%grp_tier(level=tier2);

%grp_tier(level=total);
```

Cleanup

Here, the code deletes the back-up spreadsheet created automatically by Excel.

```
data _null_;
  call system("rm &tdir./bws_waterfall_data.xlsx.bak");
run;
```

Note: &tdir was set earlier in the program with a %let statement.

```
%let tdir = /usr04/ap/analyst/jd74575/1_miscellaneous/output;
```

Next, the code creates a preferred backup by copying the data spreadsheet to another spreadsheet with today's date in the name.

```
data _null_;
  call system("cp &tdir./bws_waterfall_data.xlsx
                  &tdir./bws_waterfall_data_&filedate..xlsx");
run;
```

Creating a list of the worksheets in the report spreadsheet

First the code generates a listing of all worksheet and columns in the spreadsheets and stores the unique worksheet names in a data set so that they can be listed in the email sent to the distribution list.

```
libname xl xlsx "&tdir./bws_waterfall_data.xlsx";

Proc contents data=xl._all_ memtype=data
  out=memlist noprint;
RUN;

proc sort data=memlist(keep=memname) nodupkey
           out=mem_unique;
  by memname;
run;
```

Use xl._all_ so that all worksheet names are written to data set.

Distribution list for email

This code reads the email distribution worksheet in the report spreadsheet and generates the email that is sent to everyone on the distribution list. The distribution list could be stored elsewhere, maybe in the spreadsheet that has all email distribution lists.

```
libname xl xlsx "&tdir./bws_waterfall_report.xlsx";

data email_info;
  set xl.distribution;
  output;
run;
```

Notice that the distribution list is in the report worksheet and not the data worksheet. That is because the report worksheet is not deleted and the data worksheet is. Also, the report worksheet could be maintained by the user and they would determine who gets the report.

Example of email distribution worksheet

Person	Email_address	Email_reference
Mary Wilson	mary.wilson@anybus.com	BCC
John Smith	john.simith@anybus.com	TO
David Jones	david.jones@anybus.com	TO
Robert James	robert.jones@anybus.com	CC

The table above describes the form and content needed for the code that follows. This code builds a tolist, a cclist, and a bcclist. Currently, it expects a maximum of five entries per list, but that can be easily be expanded by changing the macro cnt variable to more than 5. And, if the length of each name averages longer than 30 characters, that can be changed by modifying the 30 in the macro lnth variable calculation. %Eval is required for all macro calculations.

The purpose of this code is to concatenate the email addresses into a single email list for each group.

Code to create the email lists.

```
%let cnt = 5;
%let lnth= %trim(%eval(&cnt*30));
data addresses;
  length tolist  $&lnth;
  length cclist  $&lnth;
  length bcclist $&lnth;
  retain tolist ' ' cclist ' ' bcclist ' ';
  keep tolist cclist bcclist;
  set email_info end=eof;
  if upcase(email_reference) = 'TO' then
    do;
       tolist = strip(tolist||' '||strip(email_address));
    end;
  else if upcase(email_reference) = 'CC' then
    do;
       cclist = strip(cclist||' '||strip(email_address));
    end;
  else if upcase(email_reference) = 'BCC' then
    do;
       bcclist = strip(bcclist||' '||strip(email_address));
    end;
  if eof then
    do;
      call symput('tolist',tolist);
      call symput('cclist',cclist);
      call symput('bcclist',bcclist);
      output;
    end;
run;

%let tolist   = %trim(&tolist);
%let cclist   = %trim(&cclist);
%let bcclist  = %trim(&bcclist);
```

Be sure and trim each macro variable to prevent blanks at end of the string stored in the variable.

Using filename to send an email.

See http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002058232.htm for a good explanation of using Filename to send email. Essentially, it allows the programmer to send email by using a simple Data _Null_ step that writes a message to the Filename reference (here: Outbox) without using HTML or by using HTML BODY with either a Data _Null_ or a Proc Print contained within the HTML BODY. This is the method shown here.

The Filename reference implements all of the email addresses and subject content. The same reference could be used multiple times to send different email content to the same persons. Only the Data _Null_ would change.

```
filename outbox email
   ct      = "text/html"
   subject= "Early Warning Waterfall Report"
   from    = "jimmy.defoor@citi.com"
   to      = "&tolist"
   cc      = "&cclist"
   bcc     = "&bcclist"
   ;
```

The ODS HTML body references the Filename established above. The RS=none option forces ODS to perform record-based output. Otherwise, each Put string is continued immediately after the last string is written. Notice that the Data _Null_ step writes to File Print and not to Output. A Proc Print would do the same by default.

```
ods html body=outbox rs=none;

Title1 "The Early Warning Waterfall Reports for &rpt_mth are available";

data _null_;
  set mem_unique;
  file print;
  if _n_ = 1 then
    do;
      put;
      put "The Early Warning Waterfall data for &rpt_mth has been created"
      put "and is located at I:\Documents\1_waterfall.";
      put;
      put 'To see the data in the Waterfall report, open the report and"
      put 'then open the source under the Edit Links option of the Data tab."
      put 'The report will then update. Close the source and save the report"
      put 'to your own directory.';
      put;
      put 'Listed below are the individual reports in the worksheet';
      put;
    end;    end;
  put @10 memname;
  put ' ';
run;

ODS HTML Close;
```

The &rpt_mth was created earlier with a Call Symput and contains the current year and month in numeric form; e,g. 201809.

When the Close is executed, the email text is sent to the Filename, which delivers the email.

This website (http://support.sas.com/rnd/web/intrnet/dispatch/ods.html) is a good reference for using ODS HTML.

Example of email content

The Early Warning Waterfall Reports for September 2018 are available
The Early Warning Waterfall data for September 2018 has been created and is located at I:\Documents\1_waterfall. To see the data in the Waterfall report, open the report and then open the source under the Edit Links option of the Data tab. The report will then update. Close the source and save the report to your own directory. Listed below are the individual reports in the worksheet. MISSING_POP REPORT_MONTH STARTING_POP TIER1 TIER2 TOTAL

Conclusion

The purpose of this paper was to show a method of creating and maintaining an Excel-based report using simple Excel techniques of and basic SAS coding methods. The Excel techniques were

- Formatted cells
- Linked cells
- ISBLANK function

The SAS coding methods were

- Proc Export
- Proc Summary
- User Formats
- Macro
- Data Step
- Filename Email
- ODS HTML
- Call System

References

Ferriola, Frank (2002). What's Your _TYPE_? How to find the CLASS You Want in Proc Summary'.
Proceedings of the Twenty-seventh Annual SAS Users Group International Conference. Cary, NC: SAS Institute, Inc. See is http://www2.sas.com/proceedings/sugi27/p077-27.pdf.

SAS/IntrNet 9.1: Application Dispatcher , The Output Delivery System.
http://support.sas.com/rnd/web/intrnet/dispatch/ods.html, Cary, NC: SAS Institute, Inc .

SAS(R) 9.2 Language Reference: Dictionary, Fourth Edition, FILENAME Statement, EMAIL (SMTP) Access Method,
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002058232.htm

Contact Information

Contact the author at:

Jimmy DeFoor
Email: jimmydefoor@gmail.com