Paper BB-10

A Paperless Report Generation and Distribution System

George P Sharrard Ph.D. GPS Corp Rowayton, CT

Abstract

All over the world, business units are looking for ways to be more "Green". What better way to enhance the "Greenness" of your projects then by offering paperless solutions for the reports you generate.

Using a web server, reports can be generated in .pdf and/or .html format for anyone in your organization to access. Need more security? Want to be more specific as to who sees your reports? Then don't use the web server – generate the same .pdf and .html reports as email attachments. If your reports take the form of daily notifications then maybe you should consider sending emails where the body of the email is nicely formatted html.

In this paper we will examine the filename statement's options as they apply to the email engine; specifically the 'to', 'subject', 'attach', and 'type' options. For web based reports we will look at using the ftp engine on the filename and highlight the use of ODS PDF and ODS HTML for creating very flexible output.

The Business Case

Each day XYZ Corp receives orders for it's products. The Directors of the Marketing and Finance departments have a simple overview they want to see each day. The Recently Entered Orders report shows all orders received in the last 3 days and the Order Statistics
report is a month-to-date count of orders and revenue. The reports are run each night by the IT department – sent to a printer – and dropped off at each recipient's office by the mail room staff. Finance, Marketing and Accounting each type specific numbers from the reports into spreadsheets to meet the specific needs of each department. It would be much better if the reports were generated in an electronic form that allowed for copy/paste functionality. Also, I want the program that creates the reports to distribute them automatically.

Passive Distribution

One way to get your reports out to the business users is simply to post them to a web server. A web server behind your companies firewall provides for a pretty good level of security (limited by the quality of your company's firewall). Talk to your IT group and get a logon to the web server machine and get the full pathname where you will be allowed to write your daily reports.

Write your SAS® code to produce the reports as specified by the Marketing and Finance departments. When you have everything working just the way you like; put the following filename 'wrapper' around the part of your program that produces the reports.

```
filename outf ftp "XYZCorp_New_Orders_&dt..html"   host='20.10.111.222'
user='username' pass='password'
cd='/pathname/finance/reports' ;

options ls=256 ps=55 pageno=1 nodate symbolgen orientation=landscape ;

* ods pdf body= outf style=D3d  fontscale=50 ;
ods html body= outf palette = fancyprinter   ;

Your report code goes here

ods html close ;
* ods pdf close ;
```

What does this do? First we are assigning the logical name "outf" to our output file. The "ftp" part invokes the SAS® ftp engine. – which does just what its name implies. Whatever you write to outf1 gets ftp'd to the IP address specified in the host= parameter. Because you are making a foreign connection to this host you must supply a username and password (user= and pass=). Unless you are storing the daily report files in your home directory [very unlikely] you need to issue a change directory command (cd=).

The reports can be created as either .html or .pdf formatted files. Use -- ODS pdf body=outf1 – to make a pdf report. .OR. Use – ODS html body=outf -- to make an html report. For a pdf report use 'style= ' to change to look of the report. For an html report use 'palette= ' to change the colors and shadings.

We include a marco variable [&dt] as part of the report name. Early in the program we would have:

```
data _null_ ;
   x1 = date() ;
   x2=put(x1,date9.) ;
   call sypmut('dt',x2) ;
run ;
```

Now we have a global macro variable showing today's date in date9. format. Reports run in the past will remain on the web server until you delete them – so you don't have to rerun reports from the past if your business users need them. Just change the date in the URL.

Active Distribution- Say Hello To The EMAIL Engine

The web server approach is akin to broadcasting, you send the information out with limited control over who views it. So, let's look at narrow casting – a system where only preselected business users will receive the daily reports

Generally, when we create files in SAS® we make SAS® datasets. The basic function of the SAS® filename statement is to read from or write to a file which is not a SAS® dataset. In display manager, when we use the Import or Export pmenu options; a filename is generated that names, gives the location of, and defines the type of the external file we want to read or write. Mostly, these files would be tab delimited (.txt) or comma separated values (.csv). To write out a 'flat file', the basic form of the filename statement is:

```
filename outf1 '../sharrard/cust_list_01.txt' ;
```

As they say on the cable TV commercials – "But wait, there's more!".
Take a look at this:

```
filename mymail email to=("george.sharrard@xyzcorp.com"
                          "Adam.Accounting@xyzcorp.com"
                          "frank.finance@xyzcorp.com"
                          "marty.marketing@xyzcorp.com")
subject="Daily Marketing Reports"
type='text/html'
attach=("../sharrard/Recently_Entered_Orders.html"
        "../sharrard/Order_Statistics.html")
;
```

mymail is the logical name of your output file.
email is the SAS® engine that defines the special nature of your output.
to, subject, type, attach
these are all features associated with the email engine

The 'to=' parameter is simply who you want your output file to be emailed to. [On a UNIX/Linix system, the email engine hooks up with SMTP to send your mail. On a desktop machine you need to have your mail program open and available.]

If you want to have text in the subject line of your email – you supply it via the 'subject=' parameter. Using double quotes allows you to enter macro variables as part of the subject line.

'attach=' allows you to have attachments on your email. [Of course, the attachments have to exist before the email is sent.]

'type=' is how you make the body of your email look more professional. By default, the email engine creates your body text as 'line printer' style output. Courier font – no colors or highlights. UGH! That will never do.

Assume we are using MS/Outlook as our email system. I type up the email the way I think it should look and send it to my boss for review. She changes it – adding colors and fonts and a fancy signature block at the end. This could be a real pain as the colors and fonts in Outlook may not be easy to recreate in SAS®. And that

signature block is more trouble than I want to think about. So, let's do it the easy way. In Outlook save the sample email as an .html file – open this file in a browser – and View Source. Everything you see is what needs to be written to the body of your email. Copy the entire html source and paste it into an empty Display Manager Program Editor window. This text is static, it will be the same everyday. Only the attachments will change.

Here is what your code would look like:

```
data _null_ ;
   file mymail ;
   put @1  "<html><BR>" /
 '<html xmlns:o="urn:schemas-microsoft-com:office:office" '/
 'xmlns:w="urn:schemas-microsoft-com:office:word"         '/
 'xmlns:st1="urn:schemas-microsoft-com:office:smarttags"  '/
 'xmlns="http://www.w3.org/TR/REC-html40">                '/
 '<head> '/
 '<meta http-equiv=Content-Type content="text/html; charset=windows-1252"> '/
 '<meta name=ProgId content=Word.Document>  '/
 '<meta name=Generator content="Microsoft Word 11">  '/
 '<meta name=Originator content="Microsoft Word 11">  '/
 '<link rel=File-List href="Aloha_files/filelist.xml"> '/
 '<title>Aloha,</title>  '/
 '<o:SmartTagType namespaceuri="urn:schemas-microsoft-com:office:smarttags" '/
 ' name="Street"/> '/
 '<o:SmartTagType namespaceuri="urn:schemas-microsoft-com:office:smarttags" '/
 ' name="State"/> '/
 '<o:SmartTagType namespaceuri="urn:schemas-microsoft-com:office:smarttags" '/
 ' name="City"/>  '/
 '<o:SmartTagType namespaceuri="urn:schemas-microsoft-com:office:smarttags" '/
```

Everything after the `put @1` is what I copied from my browser window. I added the quotes and new line symbol (/) to each line. Yes, it is a lot of tedious typing but it produces exactly what your boss asked for. Notice, on the second line the file statement references mymail which is the logical name assigned to your filename statement which is using the email engine. So, everything following the `put @1` statement will show up in the body of your email.

Of course, what shows up in your first test email is all the html code, unresolved. One very special command - type='text/html' – is what makes Outlook resolve the html code and produce a very professional looking email.

The bulk of the program is still all about the daily reports. As we did when making web reports; use ODS PDF or ODS HTML in conjunction with a filename to create a report file. You reference these files using:

```
attach=("../sharrard/Recently_Entered_Orders.html"
        "../sharrard/Order_Statistics.html")
```

You can also include daily "notices" in the body of the email. For example, if you wanted to list the 5 top selling XYZCory products for that day you would change the `data _null_` to include a set statement. First, take the daily order file used to create the attached reports. Sort it by product number and summarize the number of each product ordered. Sort the summarized file descending by 'number of units ordered'. Make a final dataset of the first 5 rows. Assume the name of this dataset is 'top5'. Now our code would read:

```
data _null_ ;
   set top5 end=lastrec ;
   file mymail ;
   if _n_ = 1 then do ;
   put @1  "<html><BR>" /
 '<html xmlns:o="urn:schemas-microsoft-com:office:office" '/
 'xmlns:w="urn:schemas-microsoft-com:office:word"         '/
 'xmlns:st1="urn:schemas-microsoft-com:office:smarttags"  '/
 'xmlns="http://www.w3.org/TR/REC-html40">                '/
'<head> '/
 '<meta http-equiv=Content-Type content="text/html; charset=windows-1252"> '/
```

The rest of the html code to make the static body text goes here

Before you close the html block simply add:

```
Put  "Today's Top Sellers" / product_no  12.  product_name $char35. units 12. ;
Else do ;
Put product_no  12.  product_name $char35. units 12. ;
If lastrec then put '</body> </html>' ;
```

Now the static text is followed by a list of the top 5 sellers for the day.

Conclusions

There is no substitute for accurate, timely reporting. Providing information that is correct is what is expected. There are no 'Gold Stars' for writing programs that are error free. What will get you noticed and provide value added benefits for your organization are the things that make the information in your reports more useful. How information gets delivered is important as is making sure that the correct people get the information.

Posting reports to a centralized web server allows for easy access. It also allows for copy/paste functionality; relieving the business user the tedious/error prone task of retyping numbers into a spreadsheet. For a more targeted distribution nothing beats email. Reports become attachments and are delivered only to the people who are on the distribution list.

The filename engines provided by SAS® give you the functionality to easily distribute reports without printing them out on paper.

Author Contact Information

George P Sharrard
GPS Corp
14 Sunwich Road
Rowayton, CT 06853
(203) 838-7248
georges00z@aol.com

Acknowledgements