

Sending Email via SAS - An Introduction and More Advanced Concepts

Bruce Nawrocki, NC Department of Health and Human Services

ABSTRACT

There are many instances when you might like to notify your co-workers when a SAS dataset or report has been updated. If you are creating these emails yourself, you might save a lot of time by writing SAS code to write these emails for you.

For example, upon successful SAS code completion, a few additional lines of SAS code can create a simple text-based email. Your email can be customized by modifying fonts and layouts. More advanced techniques can be used to write SAS procedure output, include images, or attach files.

INTRODUCTION

We'll start with a simple example of writing SAS code that creates an email, and then add complexity as we go along. The SAS code writes the email "header" and "body" to a "file" that is sent to your outgoing SMTP mail server.

SEND A SIMPLE EMAIL TO ONE PERSON

To send an email, you need at least three parts to your SAS code:

- An **OPTIONS** statement to tell SAS where to find your email server.
- A **FILENAME** statement to associate a *file-ref* with your email server, and to set the header information of your email – from-address, to-address, subject, etc.
- A **DATA** step to create the email with the above header, and to provide the email body.

Putting together these three parts, you get something like this, which will send a "test" email to yourself. The only parts you need to change are shaded in gray:

```
options emailsys=smtp emailhost="my-email-server-name" emailport=25;
filename outbox email
  from="my-email@mycorp.com"
  to="my-email@mycorp.com"
  subject="Test";
data _null_;
  file outbox;
  put "This is a test";
run;
```

Just a few words of explanation:

- The **OPTIONS** statement only needs to be submitted once. Typically, options stay set for the duration of your SAS session.
 - The first time you write this **OPTIONS** statement, the most difficult part of this code is determining your **outgoing SMTP email server name**. Sometimes you can view this under your email software's settings. Our Outlook software is locked down, and you need Admin rights to see this. In that case, you would need to ask someone in your IT group for the outgoing SMTP email server name (which may be different from the "incoming" server name).
 - **EmailPort** is typically 25, which is the SAS default, so no need to specify EmailPort unless the port is different.

- The FILENAME statement's *file-ref* – here it's called OUTBOX – can be any word up to 8 letters long. You must reference the "email" option here.
- The DATA step creates a **_null_** dataset, since you don't usually need to save the data created here to a real SAS dataset file in a folder – you just need a DATA step if you're planning to write DATA step statements, such as PUT. This is where you write the body of your email – in this case, it's just the line "This is a test".

WHAT ABOUT CC AND BCC RECIPIENTS?

Even if you don't plan to send your email to any CC or BCC recipients, you can still use add placeholders for them, like this:

```
filename outbox email
  from="my-email@mycorp.com"
  to="my-email@mycorp.com"
  cc=
  bcc=
  subject="Test";
```

SEND EMAIL TO MULTIPLE PEOPLE

You can send an email to multiple people by writing a list of names in the FILENAME statement.

Your email to-list might consist of a space-delimited list of email addresses. Each email address should be within quotes. The entire list is surrounded by a one set of parentheses. Use the same list format for your optional cc-list and bcc-list.

```
filename outbox email
  from="my-email"
  to=("emailSue" "emailTom")
  cc=("emailMyBoss" "emailTomsBoss")
  bcc=("emailCEO" "emailCIO")
  subject="Test";
```

SENDING AN ATTACHMENT

If you want to attach a single file, you can reference it in the FILENAME statement, like this:

```
filename outbox email
  from="my-email"
  to=("emailSue" "emailTom")
  cc=
  bcc=
  attach="C:\TEMP\MYFILE.TXT"
  subject="Test";
```

You can attach a list of files this way:

```
attach=("C:\TEMP\June2024.txt" "C:\TEMP\July2024.txt")
```

You can attach any type of file. SAS tries to determine the filetype based on the filename. For example, SAS will assume this attached file is an Excel spreadsheet:

```
attach="C:\TEMP\CountyPopulation.xlsx"
```

SENDING A TEXT TO A CELLPHONE

Sending a text message to a cellphone works like sending an email message. Most cellphone companies have set up an email address that functions as a gateway to your phone's text messages. If you know the recipient's cellphone company and phone number, you can determine their phone's SMS "email" address using this table. Replace "number" with your 10-digit cellphone number.

Cellphone Company	Email address for SMS text
AT&T	number@txt.att.net
T-Mobile	number@tmomail.net
Verizon	number@vtext.com

Table 1: Cellphone Company Email Addresses

For more cellphone companies, and up-to-date information, ask your favorite Web search tool for "SMS via email".

If you do decide to send a text message, make it simple and brief – SMS texts are usually limited to 160 characters. If your text is too long, or has attached files, it likely won't go through to the cellphone.

MODIFYING THE BODY OF THE EMAIL

As seen in the simple example above, you can write text into the body of the email by using PUT statements. Here's an example writing two lines of text:

```
data _null_;  
  file outbox;  
  put "This is a test";  
  put "This is the second line of text";  
run;
```

By default, the email body is sent as "text/plain" content type. That means, if you try to incorporate some HTML tags within your body text, such as and for bold text, your email body will show the HTML tags as-is (literally and), and will NOT bold your text as expected.

```
data _null_;  
  file outbox;  
  put "This is a test";  
  put "<b>This is the second line of text</b>";  
run;
```

Email body will look like this:

```
This is a test  
<b>This is the second line</b>
```

If you want to incorporate HTML tags into the body of your email, add the TYPE= option to your FILENAME statement, and specify "text/html", like this:

```
filename outbox email
  from="my-email"
  to="bossEmail"
  cc=
  bcc=
  subject="Test"
  type="text/html";
```

Now, if you re-submit this DATA step:

```
data _null_;
  file outbox;
  put "This is a test";
  put "<b>This is the second line of text</b>";
run;
```

Email body will look like this:

This is a test **This is the second line**

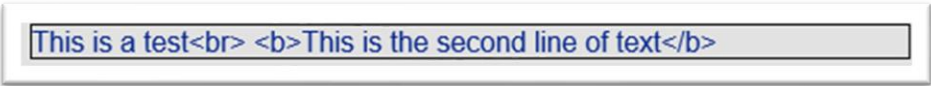
Hmm. Those two lines of text were written to the same line! This brings up an important point: If your email body is "text/html", you will need to specify where line-breaks occur, by adding line-break HTML tags, such as
:

```
data _null_;
  file outbox;
  put "This is a test<br>";
  put "<b>This is the second line of text</b>";
run;
```

Perhaps you prefer ODS over using a DATA step with PUT statements. Fine, let's try that. If you know ODS, you might know you need to surround your text with ODS HTML (since the email is composed of HTML output). I added an ID= option, so we're only opening and closing the "mail" HTML, not the main SAS HTML output.

```
ods html (id=email) body=outbox;
ods text = "This is a test <br>
          <b>This is the second line of text</b>";
ods html (id=email) close;
```

This gives us this email text:



This is a test
 This is the second line of text

But that's not right, is it? The HTML tags are being ignored again! That's because our Outlook email server is stripping the HTML tags' meaning, and just passing them back to you as-is. Also, there's a box drawn around the text.

You might try replacing HTML with HTML5, like this:

```
ods html5 (id=email) body=outbox;  
ods text = "This is a test <br>  
          <b>This is the second line of text</b>";  
ods html5 (id=email) close;
```

In the output from this code, the box around your text has been removed, but the HTML tags are still ignored:

A screenshot of an email body showing the text "This is a test
 This is the second line of text". The text is displayed in a blue font and is enclosed in a thin rectangular box.

To remedy that, we can revert back to HTML and use PROC ODSTEXT instead:

```
ods html (id=email) file=outbox;  
proc odstext;  
  p "This is a test";  
  p "<b>This is the second line of text</b>";  
run;  
ods html (id=email) close;
```

The output now looks like this, which is what we want.

A screenshot of an email body showing the text "This is a test" followed by "This is the second line of text" in bold. The text is displayed in a black font and is enclosed in a thin rectangular box.

You might also want to try replacing HTML with HTML5, like this:

INCORPORATING SAS ELEMENTS INTO EMAIL BODY

SAS MACRO VALUES

Rather than an email body with fixed text, perhaps you'd like to add dynamic text to it. Perhaps you'd like to add a SAS macro value. Let's create a macro value to store today's date:

```
%let todaysDate = %sysfunc(today(), mmddyy10.);
```

Let's also create a SAS macro value that contains the number of records in a dataset. In this example, we're counting the number of cars in the sashelp.cars SAS dataset and storing it into a macro value named totalCars.

```
proc sql;  
  select count(*) into :totalCars  
  from sashelp.cars;  
quit;
```

We can use the above two macro values in the body text of our email like this. Remember that, as with any macro values within a quoted string, you must use double-quotes, not single-quotes.

```
data _null_;  
  file outbox;  
  put "On &today'sDate, we have <b>&totalCars cars</b> on our lot.";  
run;
```

Email contains this line:

On 09/23/2024, we have **428 cars** on our lot.

SENDING SAS PROC OUTPUT TO EMAIL BODY

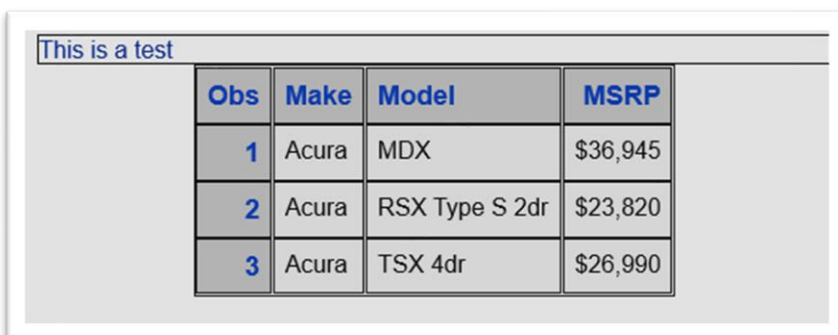
You can include SAS output – procedure reports and charts, for example – in the body of your email. Perhaps you want to include a report showing the first 3 records in the **sashelp.cars** SAS dataset. You could use this PROC PRINT code to generate the report:

```
proc print data=sashelp.cars (obs=3);  
  var make model MSRP;  
run;
```

How would you incorporate that report output into your email body? In your SAS code, insert your PROC PRINT step within ODS statements, referencing the file-ref you defined in the FILENAME statement. Unfortunately, although your email text type is “text/html”, you cannot use ODS HTML. Instead, use ODS TAGSETS.MSOFFICE2K, which is a built-in SAS style template.

```
ods tagsets.msoffice2k body=outbox;  
  
ods text = "This is a test";  
  
proc print data=sashelp.cars (obs=3);  
  var make model MSRP;  
run;  
  
ods tagsets.msoffice2k close;
```

Output looks like this:



This is a test			
Obs	Make	Model	MSRP
1	Acura	MDX	\$36,945
2	Acura	RSX Type S 2dr	\$23,820
3	Acura	TSX 4dr	\$26,990

As an aside, Chevell Parker at the SAS Institute has created an updated tagset, MSOFFICE2K_X, with additional formatting options – see References below.

The output above uses a default style. You can override the style with any other SAS pre-defined style – for example, **seaside**:

```
ods tagsets.msoffice2k body=outbox style=seaside;

proc print data=sashelp.cars(obs=3);
  var make model MSRP;
run;

ods tagsets.msoffice2k close;
```

This is a test			
Obs	Make	Model	MSRP
1	Acura	MDX	\$36,945
2	Acura	RSX Type S 2dr	\$23,820
3	Acura	TSX 4dr	\$26,990

If you want to choose a different style, you can view SAS pre-defined styles by entering **odstemplates** on the SAS command line. Double-click (or open) Sashelp.Tmplmst then Styles to view the templates that SAS provides, including “seaside”.

Perhaps there’s an existing SAS template you like, but you want to override some of its properties. You can modify an existing SAS template using PROC TEMPLATE. For example, we use a template based on “seaside”, but remove the outer borders, using this code:

```
ods path(prepend) work.template(update);
proc template;
  define style styles.ourStyle;
    parent=styles.seaside;
    class body / htmlstyle="border:none";
    class systitleandfootercontainer / htmlstyle="border:none";
    class page / htmlstyle="border:none";
  end;
run;
```

Then we reference our custom style – ourStyle – in the ODS TAGSSETS line:

```
ods tagsets.msoffice2k (id=email) file=outmail style=ourStyle;
```

CONCLUSION

Writing SAS code is great way to create automated dynamic emails, especially if they are generated on a regular basis and tend to follow a specific template.

REFERENCES

Chevell Parker's tagset definition, MSOFFICE2K_x. Accessed August 23, 2024.
<https://github.com/sascommunities/technical-support-doc/blob/main/msOffice2kTagset.md>

SAS Blog, Chris Hemedinger, "How to send a text message with SAS".
<https://blogs.sas.com/content/sasdummy/2016/08/12/how-to-send-a-text-message-with-sas/>

SAS online webinar, Chris Hemedinger, "Tips and Tricks for Sending Emails Using SAS".
https://www.sas.com/en_us/webinars/tips-tricks-sending-email.html

CONTACT INFORMATION

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

Bruce Nawrocki
NC Department of Health and Human Services
bruce.nawrocki@dhhs.nc.gov