

Sending E-Mail From a Mainframe Using SAS® in an MVS Environment

Michelle Gillespie, Louisiana State University, Baton Rouge, LA

Doug Pacas, Louisiana State University, Baton Rouge, LA

ABSTRACT

Over 75,000 financial aid checks are disbursed to LSU students each year. In an effort to better serve the student body, a SAS program has been created to send e-mails that notify students when financial aid is posted to their accounts. This automated communication has significantly reduced the heavy volume of calls previously received by campus offices regarding the timing of incoming financial aid moneys. The simple process uses check detail information and e-mail addresses to create an output dataset which is transferred to an SMTP mail server on MVS.

INTRODUCTION

Businesses and universities are often faced with the need to inform large numbers of clients quickly and cost efficiently. On the LSU campus, over 75,000 financial aid and scholarship checks are disbursed each year through a mainframe system. The timing of these disbursements is dependent upon LSU first receiving money from the source that granted the award. Although the anticipated dates of these transactions are published, the actual posting dates may vary. Previously, large numbers of students phoned the university multiple times each semester to check on the status of their accounts. In an effort to better serve the student body and to reduce the work load for LSU employees, SAS is now used to send a personalized e-mail notification to each student when financial aid is posted to his or her account. Automated communication allows information to be disseminated efficiently, significantly reducing the number of students contacting the university by phone. This paper presents details of two SAS programs which can be used together to generate a mass e-mailing of personalized messages.

WHY USE SAS®?

SAS is a natural choice for addressing an e-mail generation problem such as the one just described. The data necessary for such a mass e-mailing (e.g., social security numbers (SSN), e-mail addresses, data specific to each individual) may be obtained from a variety of sources. The power of SAS to input, sort, merge and output this data makes it preferable to other mainframe programming languages.

The programs described in this paper were coded several months ago using SAS® version 6. There are new and exciting capabilities for a direct e-mail system interface coming in SAS® version 8. We look forward to exploring the new technology upon implementation of this newest version at our site.

THE CONCEPT

The concept behind the mass e-mailing of personalized messages is quite simple. SMTP (Simple Mail Transfer Protocol) is a started task in MVS which reads in a dataset of batched e-mails and consequently transmits each e-mail message to the appropriate address. The first SAS program described in this paper outputs a dataset of batched e-mail messages. Because it is possible to overload SMTP, a second program is used to divide this large dataset into multiple small datasets before sending the e-mails to SMTP.

PROGRAM I – CREATE A DATASET OF BATCHED E-MAIL MESSAGES

```
CALL SYMPUT('LASTHDR', NUMHDR);
RUN;
```

IDENTIFY THE POPULATION

As in any mass mailing, it is necessary first to identify the population to which the correspondence will be sent. In this application, a dataset of financial aid grants and loans that have been awarded to LSU students is read by the program. Multiple awards per individual may exist. Therefore, the SAS program sorts this dataset by social security number.

```
DATA SSNS;
  INFILE SENDTOS;
  @1      SSN          $CHAR9.;
  @10     PGMCODE      $CHAR4.;
  @14     PGMDESC      $CHAR30.;
  @44     SEMESTER     $CHAR13.;
  @57     CHKAMT       ZD7.2;
```

```
PROC SORT DATA = SSNS;
  BY SSN;
```

RETRIEVE THE E-MAIL ADDRESS

LSU maintains a DB2 table of current e-mail addresses for students, uniquely keyed by social security number. A DB2EXT procedure is executed in the SAS program to extract current e-mail addresses for all students. The extracted list is then sorted by social security number and a SAS MERGE is performed to associate each student's e-mail address with his or her financial aid data. This merged dataset will serve as the driver in generating the dataset of batched e-mail messages.

```
PROC DB2EXT OUT=PWSDATA;
  SELECT      DESKTOP_ID,
              PRIMARY_ACCESS_ID
  FROM        PWS.DESKTOP;
  RENAME      1=SSN
              2=ACCESSID;
```

```
PROC SORT DATA = PWSDATA;
  BY SSN;
```

```
DATA SSNSPWS;
  MERGE PWSDATA (IN=A)
        SSNS    (IN=B);
  BY SSN;
  IF A AND B;
```

INPUT COMMON HEADER AND TRAILER DATA

The body of the message to be sent to each student in this process consists of three components. A common header paragraph informs the student that financial aid activity has been posted to his or her account. This is followed by personalized information regarding each financial aid award. A common trailer paragraph concludes the body providing information on when the student can expect to receive a mailed check. The header and trailer paragraphs are stored in separate datasets and are read into the SAS program, while the financial aid data specific to the individual is stored in the merged dataset previously described.

```
DATA HEADER;
  INFILE HEADER END=EOF;
  INPUT @1 MSGLINE $CHAR80.;
  NUMHDR + 1;
  IF EOF THEN
```

```
DATA TRAILER;
  INFILE TRAILER END=EOF;
  INPUT @1 MSGLINE $CHAR80.;
```

```

NUMTLR + 1;
IF EOF THEN
    CALL SYMPUT ('LASTTLR', NUMTLR);
RUN;

```

ASSEMBLE THE E-MAILS

The objective of the SAS program is to generate a dataset that can be fed to an SMTP mail server. Therefore, the records in this dataset must conform to the standards understood by SMTP within the MVS environment. Accepted format standards for sending SMTP e-mail have been published as "Request for Comments" numbered RFC821 and RFC822. Below is a sample of data records which would be output to a dataset by this SAS application and then consequently fed to SMTP.

```

HELO LSUMVS.SNCC.LSU.EDU
MAIL FROM:<BURSAR@LSUMVS.SNCC.LSU.EDU>
RCPT TO:<student1@lsu.edu>
DATA
Date: 26MAY2001 0:15:14 CDT
From: <BURSAR@LSUMVS.SNCC.LSU.EDU>
ReplyTo: <BURSAR@LSU.EDU>
To:<student1@lsu.edu>
Subject: Financial Aid
(blank line)
(body of letter goes here)
.
RSET
HELO LSUMVS.SNCC.LSU.EDU
MAIL FROM:<BURSAR@LSUMVS.SNCC.LSU.EDU>
RCPT TO:<student2@lsu.edu>
DATA
Date: 26MAY2001 0:15:14 CDT
From: <BURSAR@LSUMVS.SNCC.LSU.EDU>
ReplyTo: <BURSAR@LSU.EDU>
To:<student2@lsu.edu>
Subject: Financial Aid
(blank line)
(body of letter goes here)
.
QUIT

```

Each e-mail message may be thought of as a message body surrounded by an inner and outer envelope. The outer envelope contains information needed to accomplish transmission and delivery including the HELO, MAIL FROM, RCPT TO, DATA, RSET and QUIT commands as outlined by RFC821. The inner envelope portion should follow format standards specified by RFC822, and consists of the required header fields DATE, FROM and TO and several optional fields including REPLY TO and SUBJECT.

The SAS program generates a file of batched e-mail messages similar to those described above by reading and processing the newly created dataset of merged financial aid information and e-mail addresses. Since this dataset is sorted by SSN, the program uses FIRST.SSN and LAST.SSN to determine when the first and last records for a given SSN are being processed. When FIRST.SSN is true, all of the commands referred to in RFC821 and RFC822 that must precede the body of the message are output. The RCPT TO: and To: commands are followed by the e-mail address associated with that SSN from the merged dataset. A blank line is then output, followed by lines of data retrieved from the header input dataset. After the common header text, one line of student award detail per award is output. When LAST.SSN is true, the common trailer portion of the body is output, followed by a line of output containing a single period in column one. If an additional e-mail is to follow, a RSET line is written out to mark the end of the current mail transmission. RFC821 commands for the next e-mail then begin. Otherwise, a QUIT line is output which signals SMTP to terminate connection and close the transmission channel.

```

ELSE READLEN = THELEN;
INPUT @1 LINE $VARYING. READLEN;
IF LINE = 'RSET'
OR LINE = 'QUIT'
THEN NUMRSET = NUMRSET + 1;

```

COMMENTS ON RFC821 AND RFC822

It is recommended that one review RFC821 and RFC822 in their entirety before writing SAS code to create the output file. However, a few specifics are worth mentioning.

- Commands, replies and host names are not case sensitive; however, mailbox user names may be case sensitive for some hosts.
- All commands and replies should be composed of characters from the ASCII character set.
- Although headers in the inner envelope must precede the message body, they are not required to appear in any particular order.
- If e-mails are being sent from a region that practices Daylight Saving Time, logic should be included to determine the proper time zone for the DATE field.
- A line containing only a period is output to indicate the end of the body. If for some reason it is desired to actually send a period in column one as part of the body, it is necessary to append a second period so that the period *within* the body will not be interpreted as the *end* of the body. The double period will be converted by the receiver host back to a single period.

PROGRAM II – DIVIDE THE E-MAIL DATASET FOR SMTP

WHY MULTIPLE DATASETS?

Theoretically, the dataset of batched e-mail messages generated by the Program I can be read and processed directly by the SMTP mail server. In reality, however, a single feed of a few hundred e-mails during peak times of the day can result in an overload. To avoid this problem, Program II is used to divide the large dataset into multiple smaller ones before being sent to SMTP. Although the SAS code in this program is more complex than in the one just described, the concept behind "dividing and conquering" the large dataset is quite simple.

DETERMINE THE OPTIMAL NUMBER OF E-MAILS PER SMALL BATCH AND THE TOTAL NUMBER OF SMALL BATCHES

How small are the smaller batches? The answer to this question may vary for each application and is dependent on internal parameters and available disk space at the time of execution. Using trial and error, we have determined that executing during late night hours and reducing the size of single feeds to 30 e-mails is sufficient to avoid overloading SMTP in our application. You may need to experiment to find an optimal batch size.

The large dataset of all batched e-mail messages is read in one line at a time. Using the RSET and QUIT commands as markers, the total number of e-mails is calculated. This total is then divided by the predetermined number of e-mails to be sent in a single batch. The result is rounded up to the next whole number and represents the total number of small batches needed.

```

%LET NUMBMAIL = 30;

DATA _NULL_;
    RETAIN NUMRSET 0;
    LENGTH LINE $200;
    INFILE PASSONE LENGTH=THELEN END=LASTONE;
    INPUT @1 PEEK $CHAR1. @;
    IF THELEN > 200 THEN READLEN = 200;

    IF LASTONE THEN DO;
        NUMBAT = CEIL(NUMRSET / &NUMEMAIL);
        CALL SYMPUT ('NUMBATCH', PUT (NUMBAT, 5.));
    END;
RUN;

```

ALLOCATE FILE NAMES

A DO loop is utilized to dynamically allocate file names OUT1 through OUTnn, where nn is the total number of small batches needed to send the entire set of e-mails to SMTP. Each file name has a destination of SMTP and SYSOUT=A for immediate processing by the mail server.

```
%MACRO XXFNAMES;
  %DO I=1 %TO &NUMBATCH;
    FILENAME OUT&I PRINTER SYSOUT=A
      DEST=SMTP RECFM=FB %STR(;)
  %END;
%MEND XXFNAMES;
```

DIVIDE E-MAILS AMONG THE SMALLER BATCHES

The input dataset of all e-mail messages is read in a second time. A macro is used to determine the appropriate output file for each line of data. RSET and QUIT lines are again used as markers to keep track of which e-mail is being processed. The e-mail number is compared to the range of numbers to be included in each of the smaller datasets. For example, e-mails 1 through 30 are output to file OUT1, e-mails 31 through 60 are output to file OUT2, and so on.

As mentioned previously, RFC821 requires that batched e-mails be separated by a RSET command. The final e-mail being sent in a transmission must be concluded by a QUIT command to close the transmission channel. Each of the smaller datasets is sent as an independent batch to SMTP. For this reason, the RSET command following the final e-mail in each batch is converted to a QUIT. A 'trick' is used to ensure that the RSET commands being evaluated are in fact RFC821 commands and not part of the RFC822 message body.

```
%MACRO XXDOFILE;
  %DO I=1 %TO &NUMBATCH;
    %LET J = %EVAL((&I-1)*&NUMEMAIL);
    %LET K = %EVAL(&I*&NUMEMAIL);
    IF &J <= RSETNUM <=&K THEN
      FILE OUT&I NOPRINT NOTITLES %STR(;)
    %END;
%MEND XXDOFILE;
```

```
DATA _NULL_;
  RETAIN          RSETNUM          00
                  NUMBAT           00
                  RFC821            1;
  LENGTH          LINE             $200;
```

```
INFILE EMAILIN END=LASTONE LENGTH=LINELEN;
INPUT @1 PEEK $CHAR1. @;
IF LINELEN > 200 THEN THELEN = 200;
  ELSE THELEN = LINELEN;
INPUT @1 LINE $VARYING. THELEN;
```

```
%XXDOFILE
```

```
IF RFC821 THEN DO;
  IF LINE = 'QUIT' THEN DO;
    DORSET = 1
    LINE = 'RSET';
  END;
  IF LINE = 'RSET' THEN DO;
    RSETNUM = RSETNUM + 1;
    IF MOD(RSETNUM,&NUMEMAIL) = 0 THEN
      DOQUIT = 1;
    IF LASTONE THEN DOQUIT = 1;
  END;
  IF LINE = 'DATA' THEN RFC821 = 0;
END;
ELSE IF LINE = '.' THEN RFC821 = 1;
```

```
IF DOQUIT THEN DO;
  NUMBAT + 1;
  PUT 'QUIT';
```

```
END;
ELSE IF DORSET THEN PUT 'RSET';
  ELSE PUT _INFILE_;
RUN;
```

RELEASE THE BATCHES TO SMTP

A final statement deallocates all of the dynamically allocated files allowing the messages to go to SMTP.

```
FILENAME _ALL_ CLEAR;
```

CONCLUSION

The basic functions of the two SAS programs described in this paper can easily be applied to other applications requiring the capability to send a large number of personalized e-mails. At LSU, a similar batch process has been developed to contact employees when supplemental checks have been directly deposited into their bank accounts. Another batch process sends e-mail notifications to individuals responsible for submitting procurement card accounting requests. In both of these cases, Program I was modified to create the master e-mail dataset. Program II was then used to divide the dataset before sending the e-mails to SMTP.

The ultimate success or failure in developing such an e-mail application is dependent upon the ability of the program to access a stable, reliable source of current e-mail addresses for the population being contacted. Therefore, every effort should be made to ensure that this data is current. Due to the fact that individuals change e-mail addresses frequently and e-mail messages are sometimes retrieved on an irregular basis, it may be advisable to develop applications that send messages as a courtesy to clients, rather than messages of a critical nature.

REFERENCES

The two SAS programs presented in this paper can be viewed in their entirety by visiting the following web site:

<http://www.lsu.edu/ocs/conference/ssu2001>

ACKNOWLEDGMENTS

Program II and key elements of Program I were designed by Frank O'Quinn, Associate Director, Technology Support Center, Louisiana State University. He has also been an invaluable resource during the writing of this paper.

CONTACT INFORMATION

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

Michelle Gillespie
Louisiana State University
Baton Rouge, LA 70803
Work Phone: (225) 578-4727
Email: mgilles@lsu.edu

Douglas Pacas
Louisiana State University
Baton Rouge, LA 70803
Work Phone: (225) 578-3718
Email: doug@lsu.edu