Paper 48-25

Using Batch MVS SAS® To Send Email Via A UNIX® Email Server

Stephen M. Englert, Dun & Bradstreet Corporation, Murray Hill, NJ

ABSTRACT

Frequently, there are customers, internal or external to a business, that require timely email notification regarding the successful completion of certain important MVS® jobs. Or, perhaps they wish to receive via email the various reports generated by these jobs. In any case, MVS is not inherently email-friendly, and, more often than not, programmers must download reports to a PC, set up a list of email recipients, scribble a quick explanatory note in the body of the email, attach the saved reports documents, and finally, press SEND. This works fine for one-off, ad hoc jobs. But, what if this level of notification is required for repetitive jobs? For example, what if a dataset is updated monthly, and certain interested parties would prefer email notification, rather than awaiting a telephone call or scrounging around the mainframe to verify the dataset's update status? How about a job that creates both statistical reports and a dataset for an external customer on a daily basis? In such cases, it would be both convenient and expedient to have the MVS job automatically set up and send the email (with or without reports) to the appropriate parties.

What follows is one possible way to resolve this issue, using Base SAS in a Batch MVS environment to set up and "drive" the movement of email text, reports, and recipient lists from MVS to an UNIX email server, which in turn sends the email to the desired parties. Set this up as a JCL PROC, and it makes a functional and fairly simple "semi-automated" email process.

INTRODUCTION

This solution will use Base SAS to accomplish most of its work. SAS is very flexible in its many ways of reading, manipulating, and writing data – and this program's simplicity doesn't even begin to sound the depths of SAS' potential. Furthermore, SAS is extremely easy to maintain and update: This program could easily be modified to address many other needs that might be specific to your IS shop.

The SAS program will:

1) Read an input file (ddname EPARMS), which will contain
   a) The UNIX Logon ID and Password that you wish to use to sign onto the UNIX email server.
   b) The list of recipients to which you want the email sent.
   c) The Subject line for your email.
   d) And any Header message that you'd like to begin your email with.
2) Read an input file (ddname REPORTIN, lrecl=80) containing any reports created in a prior job stream or job step, that you'd like to include in the email (for example, output from PROC FREQ's).
3) Create a file (ddname RECIPOUT) containing the list of email addresses for the desired Recipients (to be FTP'ed to the UNIX system).
4) Create a file (ddname RPTOUT) containing the Header and Report text combined: the body of the email (to be FTP'ed to the UNIX system).
5) Create a file (ddname EXFTPSC) containing the text for the UNIX script that will put all of this together and execute the command to send the email (to be FTP'ed to the UNIX system).
6) Create a file (ddname EXFTP) containing the commands to FTP the Report (4. Above), UNIX Email Script (5. Above), and Recipients List (1. Above) files from MVS to UNIX.
7) Create a file (ddname EXSCRIPX) containing the command to make the script (5. Above) executable by anyone.
8) Create a file (ddname EXSCRIPT) containing the command to execute the UNIX email script.

Once this has been accomplished, this process will execute the MVS utility program, IKJEFT01, thrice:

A) Execute the code in EXFTP (6. Above). This will FTP the Report (4. Above), UNIX Email Script (5. Above), and Recipients List (1. Above) files from MVS to UNIX.
B) Execute the UNIX command in EXSCRIPX to make the script just FTP'ed executable to all.
C) Execute the UNIX command in EXSCRIPT to execute the email script (send the email).

SAMPLE JOB EXECUTION

Perhaps the best way to approach this is to first take a look at a sample job. Let's assume that we have a process that runs daily, providing to the customer an email containing some explanatory text and the output from a simple PROC FREQ. The first step of our job will create the FREQ report, saving it to a catalogued dataset. Then, we'll send the email, using a catalogued JCL Proc that we'll call "EMAIL". The whole job stream might well look like this:

THE MVS JCL JOB STREAM

```
1.  //JOBNAME1  JOB ('9999-999999-99-Z-99999999999'),
2.  //      'S.ENGLERT',CLASS=I,
3.  //      MSGCLASS=H,NOTIFY=MVSID
4.  //*-------------------------------------------------SET
5.  //SET1     SET HLQ=MVSID,
6.  //         CUST=EMAILTST,
7.  //         RPTFILE=MVS.REPORT.FILE,
8.  //*        RPTFILE=NULLFILE,
9.  //         PARMS=MVSID.PDS.PDATA(EPARMS)
10. //*-------------------------------------------------UNCAT
11. //UNCATX   EXEC PGM=IEFBR14,COND=(0,NE)
12. //DD1      DD  DSN=&RPTFILE.,
13. //         DISP=(MOD,DELETE,DELETE)
14. //*-------------------------------------------------FREQRPT
15. //FREQRPT  EXEC    SAS
16. //SAS.WORK   DD  SPACE=(CYL,(200,200),RLSE),
17. //           UNIT=SYSDA
18. //SAS.SORTWK01 DD  UNIT=SYSDA,
19. //           SPACE=(CYL,(10,10))
20. //IN1     DD  DSN=MVS.RAW.DATAFILE,DISP=SHR
21. //FT12F001 DD  DSN=&RPTFILE.,
22. //         DISP=(NEW,CATLG,DELETE),
23. //         UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
24. //         DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
25. //SYSIN    DD  *
26. OPTIONS LS=80;
27. DATA IN1;
28.     INFILE IN1;
29.     INPUT @1  CURRSCOR   1.
30.               PREVSCOR   1.;
31. RUN;
32. PROC FREQ DSATA=IN1;
33.     TABLES PREVSCORE*CURRSCOR /LIST;
34. RUN;
35. /*
36. //MAIL1  EXEC EMAIL
```

Fig. 1

That's it! On the surface, that's all that needs to be done.

To merely call the JCL Proc, without any prior steps, it gets even simpler:

```
1.  //JOBNAME1  JOB ('9999-999999-99-Z-99999999999'),
2.  //      'S.ENGLERT',CLASS=I,
3.  //      MSGCLASS=H,NOTIFY=MVSID
4.  //*-------------------------------------------------------SET
5.  //SET1    SET HLQ=MVSID,
6.  //        CUST=EMAILTST,
7.  //        RPTFILE=MVS.REPORT.FILE,
8.  //*       RPTFILE=NULLFILE,
9.  //        EPARMS=MVSID.PDS.PDATA(EPARMS)
10. //*-------------------------------------------------------UNCAT
11. //MAIL1  EXEC EMAIL
```

Fig. 2

Note that lines 5-9 of the JCL are a SET statement. By using the SET statement, we can set "global" parameters for an entire JCL job stream, regardless whether or not the parameters are needed by a Proc or any other step in the stream. The following explains the parms being set in the SET statement:

HLQ: the High-Level Qualifier you wish to begin catalogued dataset names with.

CUST: a mid-level qualifier used to name catalogued datasets in this job stream.

RPTFILE: the Dataset name (DSN) that contains a Report that you want included in the email. Note that by commenting line 7 out and instead using line 8, the EMAIL Proc would receive a dataset name of NULLFILE (no report desired or available).

EPARMS: dataset name or library's member name where the EPARMS file can be found.

THE REPORT

Now, consider this example: Assume that the file containing the report, MVS.REPORT.FILE, contains the output of a SAS program that created some simple statistics for a customer, and that this file's contents look like the following:

PREVSCOR	CURRSCOR	FREQUENCY	PERCENT
	0	24663	0.1
	1	3692	0.0
	2	17344	0.1
	3	106702	0.5
	4	189335	0.9
	5	8191	0.0
0		655767	3.0
0	0	7279324	33.2
0	1	149	0.0
0	2	752	0.0
0	3	1472	0.0
0	4	3425	0.0
0	5	786	0.0
1		8473	0.0
1	0	1488	0.0
1	1	463226	2.1
1	2	95841	0.4
1	3	17146	0.1
1	4	2690	0.0
1	5	130	0.0
2		62646	0.3
2	0	12212	0.1
2	1	91876	0.4
2	2	2090345	9.5

… etc.

Fig. 3

EPARMS: THE EMAIL PARM FILE

And, assume that the file containing the EPARMS, MVSID.TSOPDS.PDATA(EPARMS), contains the following:

```
KEY FIELD          = User-Defined Values (begin in Col 20)
------------ =    ------------------------------------------------------
UNIX LOGON ID      = myunixid
UNIX PASSWORD      = myunixpw
HI LEVEL QUAL      = MVSID
CUSTOMER           = EMAILTST

RECIPIENT          = recip1@webaddr1.com
RECIPIENT          = recip2@webaddr2.com

SUBJECT            = Auto-Email TEST!!!!!

HEADER             = To All:
HEADER             = This is a sample Header.  It can be
HEADER             = multiple lines long, and contain
HEADER             = whatever you wish.
HEADER             =
HEADER             =   Thanks,
HEADER             =       The SAS Programmers
```

Fig. 4

EXPLANATION OF EPARMS

The first two lines of EPARMS simply describe the column locations in the file (line 1) and delineate a field's length (line 2: the lines of dashes). This makes filling in the file more user-friendly. VALUES can be up to 50-bytes long.

The SAS program, EMAILSAS, will read this file, looking for the KEY FIELDs to tell it where to place the VALUES in various output files (the Script, the body of the email, etc.).

UNIX LOGON ID and UNIX PASSWORD are both required fields, in order to sign onto the UNIX Email Server remotely (while still in MVS) – to FTP files and so on. Rather than have these two security-sensitive fields located in what might possibly be an easily accessed EPARMS file on your mainframe (this is a RACF issue), one might write a separate SAS/AF program to serve as a graphical user-interface that could query the user for ID and Password. The SAS/AF program could then place the ID and Password into a temporary file that will be destroyed at the end of the job's execution, pass that file to the EMAILSAS program, and even submit the JCL. For our purposes, though, ID and Password are in the EPARMS file.

HI LEVEL QUAL is one of two required Key fields (CUSTOMER is the second) that SAS will use to create literal dataset names that must be placed into the code files that will drive the FTP processes (moving the Reports file to UNIX, etc.).

CUSTOMER is one of two required Key fields (HI LEVEL QUAL is the other) that SAS will use to create literal dataset names that must be placed into the code files that will drive the FTP processes (moving the Reports file to UNIX, etc.). CUSTOMER is also used to create unique destination filenames on UNIX (for the email Script, the Recipients list, and body of the email).

RECIPIENT is a required Key field whose Value will be the email address for a single desired Recipient. There should be one RECIPIENT line in EPARMS for each email address required.

SUBJECT is a required Key field whose Value contains the text for the Subject line of the email. Only one line allowed.

HEADER is an optional Key field that will contain any descriptive, personalized or standardized notes you'd like added to the email. These HEADER comments will be placed in the body of the email, immediately prior to any Reports being inserted.

SAMPLE EMAIL OUTPUT

Given these two inputs to Proc EMAIL (the Report and the EPARMS), the email sent would look like this:

To All:
This is a sample Header. It can be
multiple lines long, and contain
whatever you wish.

Thanks,
The SAS Programmers

PREVSCOR	CURRSCOR	FREQUENCY	PERCENT
	0	24663	0.1
	1	3692	0.0
	2	17344	0.1
	3	106702	0.5
	4	189335	0.9
	5	8191	0.0
0		655767	3.0
0	0	7279324	33.2
0	1	149	0.0
0	2	752	0.0
0	3	1472	0.0
0	4	3425	0.0
0	5	786	0.0
1		8473	0.0
1	0	1488	0.0
1	1	463226	2.1
1	2	95841	0.4
1	3	17146	0.1
1	4	2690	0.0
1	5	130	0.0
2		62646	0.3
2	0	12212	0.1
2	1	91876	0.4
2	2	2090345	9.5

… etc.

Fig. 5

JCL PROC "EMAIL" (EXPANDED)

Now, let's take a look at the EMAIL Proc, expanded, so that we can match the JCL code with the general flow discussed above. Following this, we'll discuss the EPARMS file. Then we can finally discuss the SAS program (EMAILSAS, in line 63 below) itself.
The JCL Proc, called EMAIL, might look like this (referencing line numbers have been placed to the left of the JCL code):

```
1.  //EMAIL    PROC
2.  //*=======================================
3.  //* AUTOMATIC EMAIL
4.  //* AUTOMATIC EMAIL
5.  //* AUTOMATIC EMAIL
6.  //*=======================================
7.  //*
8.  //* SETUP FOR FTP AND EMAILING OF REPORTS
9.  //*
10. //*-------------------------------------------------------UNCAT
11. //UNCATX   EXEC PGM=IEFBR14,COND=(0,NE)
12. //DD1      DD  DSN=&HLQ..&CUST..RPT,
13. //           DISP=(MOD,DELETE,DELETE)
14. //DD2      DD  DSN=&HLQ..&CUST..RECIPS,
15. //           DISP=(MOD,DELETE,DELETE)
16. //DD3      DD  DSN=&HLQ..&CUST..MAIL,
17. //           DISP=(MOD,DELETE,DELETE)
18. //*
19. //SETUP    EXEC    SAS
20. //SAS.WORK     DD  SPACE=(CYL,(200,200),RLSE),
21. //           UNIT=SYSDA
22. //SAS.FT11F001 DD  SYSOUT=*
23. //SAS.FT12F001 DD  SYSOUT=*
24. //SAS.FT13F001 DD  DUMMY,DCB=BLKSIZE=80
25. //SAS.FT14F001 DD  DUMMY,DCB=BLKSIZE=80
26. //SAS.FT22F001 DD  SYSOUT=*
27. //SAS.SORTWK01 DD  UNIT=SYSDA,
28. //           SPACE=(CYL,(10,10))
29. //SAS.SORTWK02 DD  UNIT=SYSDA,
30. //           SPACE=(CYL,(10,10))
31. //SAS.SORTWK03 DD  UNIT=SYSDA,
32. //           SPACE=(CYL,(10,10))
33. //SAS.SORTWK04 DD  UNIT=SYSDA,
34. //           SPACE=(CYL,(10,10))
35. //*--------------FILE CONTAINING REPORT (LRECL=80)
36. //REPORTIN  DD  DSN=&RPTFILE.,DISP=SHR
37. //*---------------EMAIL PARMS (SIGNON, SUBJECT, ...)
38. //EPARMS       DD  DSN=&EPARMS.,DISP=SHR
39. //EXFTP        DD  DSN=&EXFTP,
40. //           UNIT=SYSDA,DISP=(NEW,PASS,DELETE),
41. //           SPACE=(CYL,(1,1),RLSE),
42. //           DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
43. //EXFTPSC      DD  DSN=&HLQ..&CUST..MAIL,
44. //           UNIT=SYSDA,DISP=(NEW,PASS,DELETE),
45. //           SPACE=(CYL,(1,1),RLSE),
46. //           DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
47. //EXSCRIPT     DD  DSN=&EXSCRIPT,
48. //           UNIT=SYSDA,DISP=(NEW,PASS,DELETE),
49. //           SPACE=(CYL,(1,1),RLSE),
50. //           DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
51. //EXSCRIPX     DD  DSN=&EXSCRIPX,
52. //           UNIT=SYSDA,DISP=(NEW,PASS,DELETE),
53. //           SPACE=(CYL,(1,1),RLSE),
54. //           DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
55. //RPTOUT       DD  DSN=&HLQ..&CUST..RPT,
56. //           UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
57. //           SPACE=(CYL,(1,1),RLSE),
58. //           DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
59. //RECIPOUT    DD  DSN=&HLQ..&CUST..RECIPS,
60. //           UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
61. //           SPACE=(CYL,(1,1),RLSE),
62. //           DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
63. //SYSIN     DD  DSN=MVSID.PDS.PDATA(EMAILSAS),
64. //            DISP=SHR
65. //*
66. //*- FTP REPORT AND EMAIL ADDRESSES TO UNIX
67. //*
68. //JEFT01  EXEC PGM=IKJEFT01,DYNAMNBR=30
69. //STEPLIB   DD DSN=MY.STEPLIB,DISP=SHR
70. //SYSPRINT  DD SYSOUT=*
71. //SYSTSPRT  DD DUMMY,DCB=LRECL=127
72. //SYSUDUMP  DD SYSOUT=*
73. //SYSIN     DD DSN=PRD.PDATA(EMPTY),
74. //           DISP=SHR
75. //SYSOUT    DD SYSOUT=*
76. //INPUT     DD DSN=&EXFTP,
77. //           DISP=(OLD,DELETE,DELETE)
78. //SYSTSIN  DD DSN=PRD.PDATA(FTPEXIT),
79. //           DISP=SHR    ** ftp (exit
80. //*
81. //*- RUN CHMOD SCRIPT TO MAKE EMAIL SCRIPT
82. //   EXECUTABLE
83. //*
84. //JEFT02  EXEC PGM=IKJEFT01,DYNAMNBR=30
85. //STEPLIB   DD DSN=MY.STEPLIB,DISP=SHR
86. //SYSPRINT  DD SYSOUT=*
87. //SYSTSPRT  DD DUMMY,DCB=LRECL=127
88. //SYSUDUMP  DD SYSOUT=*
89. //SYSIN     DD DSN=PRD.PDATA(EMPTY),
90. //           DISP=SHR
91. //SYSOUT    DD SYSOUT=*
92. //INPUT     DD DUMMY
93. //SYSTSIN   DD DSN=&EXSCRIPX,
94. //           DISP=(OLD,DELETE,DELETE)
```

```
95.  //*
96.  //*- RUN SCRIPT ON UNIX TO EMAIL THE REPORTS
97.  //*
98.  //JEFT02  EXEC PGM=IKJEFT01,DYNAMNBR=30
99.  //STEPLIB   DD DSN=MY.STEPLIB,DISP=SHR
100. //SYSPRINT  DD SYSOUT=*
101. //SYSTSPRT  DD DUMMY,DCB=LRECL=127
102. //SYSUDUMP  DD SYSOUT=*
103. //SYSIN     DD DSN=PRD.PDATA(EMPTY),
104. //             DISP=SHR
105. //SYSOUT    DD SYSOUT=*
106. //INPUT     DD DUMMY
107. //SYSTSIN   DD DSN=&EXSCRIPT,
108. //             DISP=(OLD,DELETE,DELETE)
109. //*
110. // PEND
```

Fig. 6

Remember that the JCL that calls this Proc is really quite short and user-friendly (see Fig. 2).

EXPLANATION OF PROC "EMAIL"

Lines 10-17 simply un-catalog the datasets created by any prior runs of this Proc, within this job stream.

Lines 19-64 contain the JCL to run the SAS program, EMAILSAS.

Lines 66-79 run the MVS utility, IKJEFT01 to FTP the Report, UNIX Email script code, and Recipients List to the UNIX system.

Lines 81-94 run the MVS utility, IKJEFT01 to change the permissions of the FTP'ed UNIX Email script so that anyone may execute the script.

Lines 96-108 run the MVS utility, IKJEFT01 to send the command to UNIX that will execute the Email script.

EMAILSAS

```
1.   OPTIONS NOCAPS;

2.   DATA REPORT;
3.        INFILE REPORTIN;
4.        INPUT @1 LINE80 $CHAR80.;
5.   RUN;

6.   DATA HEADER RECIPS;
7.        INFILE EPARMS;
8.        INPUT     @1  FLAG13  $CHAR13.
9.                  @20 LINE8    $CHAR8.
10.                 @20 LINE50  $CHAR50.;
11.       IF FLAG13 = 'UNIX LOGON ID' THEN
12.            CALL SYMPUT('LW',TRIM(LEFT(LINE8)));
13.       IF FLAG13 = 'UNIX PASSWORD' THEN
14.            CALL SYMPUT('PW',TRIM(LEFT(LINE8)));
15.       IF FLAG13 = 'SUBJECT'       THEN
16.            CALL SYMPUT('SB',TRIM(LEFT(LINE50)));
17.       IF FLAG13 = 'HI LEVEL QUAL' THEN
18.            CALL SYMPUT('QUAL',TRIM(LEFT(LINE8)));
19.       IF FLAG13 = 'CUSTOMER     ' THEN
20.            CALL SYMPUT('CUST',TRIM(LEFT(LINE8)));
21.       IF FLAG13 = 'HEADER       ' THEN DO;
22.            LINE80 = LINE50;
23.            OUTPUT HEADER;
24.       END;
25.       IF FLAG13 = 'RECIPIENT    ' THEN DO;
26.            LINE80 = LINE50;
27.            OUTPUT RECIPS;
28.       END;
29.  RUN;
30.
31.  ***CREATE REPORT (HEADER + REPORT FILE) ;
32.  DATA _NULL_;
33.       SET       HEADER
34.                 REPORT;
35.       FILE RPTOUT;
36.       PUT @1 LINE80 $CHAR80.;
37.  RUN;
38.
39.  ***CREATE RECIP FILE (EMAIL ADDRESSES);
40.  DATA _NULL_;
41.       SET RECIPS;
42.       FILE RECIPOUT;
43.       PUT @1 LINE80 $CHAR80.;
44.  RUN;
45.
46.  ***CREATE     1 - EXSCRIPT:  RUN EMAIL SCRIPT;
47.  ***                ON UNIX SERVER;
48.  ***            2 - EXSCRIPX:  CHMOD EMAIL;
49.  ***                SCRIPT TO EXEC ;
50.  ***            3 - EXFTPSC:   CREATE UNIX EMAIL;
51.  ***                SCRIPT;
52.  ***            4 - EXFTP:     FTP *.DAT, *_LIST,;
53.  ***                AND _SCRIPT -*;
54.  DATA _NULL_;
55.       FILE EXSCRIPT;
56.       PUT @1  'rexec -l '
57.               "&LW."
58.               ' -p '
59.               "&PW."
60.               '  UNIX  -'
61.               /
62.               "&CUST._script "
63.               ""'
64.               "&SB."
65.               ""';
66.       FILE EXSCRIPX;
67.       PUT @1  'rexec -l '
68.               "&LW."
69.               ' -p '
70.               "&PW."
71.               '  UNIX  -'
72.               /
73.               "chmod 777 &CUST._script ";
74.       FILERPT =      """ ]]
75.                      TRIM("&QUAL.") ]]
76.                      '.' ]]
77.                      TRIM("&CUST.") ]]
78.                      ".RPT"";
79.                      FILERCP = """ ]]
80.                      TRIM("&QUAL.") ]]
81.                      '.' ]]
82.                      TRIM("&CUST.") ]]
83.                      ".RECIPS'";
84.                      FILESCR = """ ]]
85.                      TRIM("&QUAL.") ]]
86.                      '.' ]]
87.                      TRIM("&CUST.") ]]
88.                      ".MAIL'";
89.       FILE EXFTPSC;
90.       PUT @1  '#! /bin/ksh'
91.               //
92.               'export charset=us\-ascii'
93.               /
94.               'export encoding=8bit'
95.               //
96.               'mailx "-s$1" '
97.               "`cat &CUST._list` < &CUST..dat";
98.       FILE EXFTP;
99.       PUT @1  'UNIX'
100.              /
101.              "&lw."
102.              /
103.              "&pw."
104.              /
```

```
105.                'ascii'
106.                /
107.                'put '
108.                FILERPT
109.                "  &CUST..dat"
110.                /
111.                'put '
112.                FILERCP
113.                "  &CUST._list"
114.                /
115.                'put '
116.                FILESCR
117.                "  &CUST._script"
118.                /
119.                'quit';
120. RUN;
```

Fig. 7

EXPLANATION OF EMAILSAS

Line 1: Set to NOCAPS, so that SAS accepts and writes mixed-case text.

Lines 2-5: Create REPORT set, containing the Reports in the REPORTIN file (LRECL=80).

Lines 6-29: Read the EPARMS file. Create:

1) Macro variables to hold the Values for various Key Fields read (lines 11-20),
2) HEADER set to hold all Header lines (lines 21-24),
3) RECIPS set to hold all Recipient email addresses (lines 25-28).

Lines 31-37: Append the REPORT to the end of the HEADER messages, and write the RPTOUT file. This will be FTP'ed to UNIX in later JCL steps.

Lines 39-44: Write the RECIPOUT file from the RECIPS set. . This will be FTP'ed to UNIX in later JCL steps.

Lines 46-120 (end):

Lines 55-65: Write to EXSCRIPT the UNIX commands that will execute the email script on the UNIX Server (see also Lines 89-97).

Lines 66-73: Write to EXSCRIPX the UNIX commands that will change user-permissions on the email script, such that anyone will be able to execute them (should the email have to be re-sent)

Lines 74-88: Create variables to hold the dataset names to be used by FTP commands. This is where the filenames are "dynamically" created, for both the input and output files to a FTP command.

Lines 89-97: Write to EXFTPSC the text of the email script (see also Lines 55-65).

Lines 98-119: Write to EXFTP the commands to FTP the Body of the email (FILERPT), Recipients list (FILERCP), and UNIX email script file (FILESCR) to the UNIX server.

As you can see, conceptually EMAILSAS is a fairly simple program. It reads a few files in, sets up a few variables (macro and other), and writes a few files out. Some of these output files are catalogued datasets that will be FTP'ed (Recipients, body of message, and email script), some are merely to support the MVS utility steps that FTP the catalogued datasets to the UNIX server or execute UNIX commands from MVS (changing permissions on the script, once it's been FTP'ed, and executing the script).

ON THE UNIX SERVER

```
/dir1/myunixid>ll EMAI*
-rw-r-----   1 myunixid   icg          486 Sep 27 09:50
EMAILTST.dat
-rw-r-----   1 myunixid   icg           81 Sep 27 09:50
EMAILTST_list
-rwxrwxrwx   1 myunixid   icg          486 Sep 27 09:50
EMAILTST_script
/dir1/myunixid>
```

Fig. 8

EMAILTST.dat contains the Header and optional Report (see Fig.5)

EMAILTST_list contains the e-mail addresses

Contents of EMAILTST_script:

```
#! /bin/ksh

export charset=us\-ascii
export encoding=8bit

mailx "-s$1" `cat EMAILTST_list` < EMAILTST.dat
```

Fig. 9

CONCLUSION

As can be seen from this example, it is possible to, in effect, send emails from directly within a MVS batch job. In this case, we used SAS to set up the body of the email, the recipients list, and the UNIX script; then FTP'ed them to the UNIX system, and effected the transmission of our email from the MVS job through the UNIX email server to selected recipients. This could easily be modified for other uses that involve MVS and UNIX cross-communications (to include remote submissions). And, for tighter security of IDs and Passwords, SAS/AF could be used to solicit user-input.

REFERENCES

ACKNOWLEDGMENTS

I wish to thank God for my tremendously supportive family, Anthony Palasciano for his help in writing this paper, Ron Klein for sharing his FTP and UNIX expertise with me, and all of my colleagues at the Dun & Bradstreet Corporation for their encouragement and assistance.

CONTACT INFORMATION

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

Stephen (Steve) M Englert

Dun & Bradstreet Corporation
3 Sylvan Way
Parsippany, NJ 07054

Work Phone: (973) 605-6646
Fax: (973) 605-6290
Email: englerts@dnb.com