Paper 002-2009

Using the DATA step as a POP3 mail client

Erik W. Tilanus, Driebergen, the Netherlands

ABSTRACT

Lasts year's presentation “Sending email from the DATA step” drew a lot of attention. But receiving email in the DATA step, i.e. using the DATA step as a POP3 mail client is also possible.
This is a simple low-cost solution for companies that require branches to send in information for handling at the head office.
This paper explains how to access a POP3 mail server and save the messages in a SAS® data set and also how to make it run repeatedly.

INTRODUCTION

Suppose you need to consolidate or process figures that are sent in from remote locations via e-mails. Can these emails be processed in SAS directly? The answer is simply: yes. And using the Socket access method in the FILENAME statement it is not very complicated.
This paper consists of three parts: first a short introduction of the POP3 protocol, then a description of a DATA step which can be used to read the mails and finally two approaches to make this DATA step run repeatedly, e.g. every 10 minutes.

POP3

A POP3 mail server is typically a TCP/IP based server. So it "listens" to a TCP/IP "socket" to start communication with a client, e.g. a mail client like Thunderbird or Outlook. The SOCKET access method which is implemented in the FILENAME statement implements TCP/IP communications and thus enables communication with POP3 mail servers. The basic format for this FILENAME statement is:

```
FILENAME pop3 SOCKET 'mail.mailserver.com:110' <other options>;
```

This version of the statement sets the SAS program in client mode and enables it to establish a two-way communication with a TCP/IP based server, listening to the same socket. In our case socket 110, which is the default socket for POP3 servers.

The POP3 protocol is in fact a very simple protocol. Each message you send to the server starts with a key word, possibly followed by some data and ending with a carriage control – line feed combination. Each response that you receive from the server starts with +OK if your message was handled successfully.

THE POP3 DIALOG

Table 1 shows you the typical sequence of messages between the client and the server. You log on to the server, check whether there are mails available and if so, you download them and delete them from the server. Finally you log off and you are done.

To start the communication with the server you issue a read request (INPUT). This is a kind of “knock on the door”. The server will respond with +OK. After that the communication sequence as outlined in Table 1 can start. If the response from the server consists of multiple lines you have to read them one by one using the INPUT statement repeatedly, until the server responds with a termination line, which is a line with a single period.

THE FULL PROGRAM

With the knowledge of the dialog it is relatively simple to write the program for receiving mails in the DATA step.

Program 1 shows a fairly complete version of a program that can do this. The FILENAME statement ① has the structure as shown above, with one additional option: TERMSTR=CRLF. As mentioned the POP3 protocol requires that each message is terminated with a carriage control (CR) and a line feed (LF) character. By default SAS terminates only with the line feed character. You could add the carriage control by adding '0D'x at the end of each PUT statement, but adding the option in the FILENAME statement is easier.

Table 1: Typical dialog with POP3 server

Action	Data	Comment
Receive	+OK	May be followed by server identification
Send	USER &myuserid	&myuserid stands for the user ID
Receive	+OK	
Send	PASS &mypassword	%mypassword stands for the password
Receive	+OK User successfully logged on	
Send	STAT	Asks for status
Receive	+OK 2 1780	2 messages present, 1780 bytes in total
Send	LIST	Ask details of messages
Receive	+OK 2 1780 1 881	First message length 881 byte (repeat receive to get info on all messages)
Send	RETR n	Retrieve n-th message
Receive	+OK followed by message text	Retrieve n-th message (repeat receive until complete message received)
Send	DELE n	Delete n-th message
Receive	+OK	
Send	QUIT	
Receive	+OK	May be followed by server identification

Program 1: Reading e-mails from a POP3 mail server

```
① FILENAME pop3 SOCKET 'pop3.planet.nl:110' TERMSTR=CRLF;
  DATA mailfile(KEEP=line);
  LENGTH line $1000;
  ARRAY msglength {100} _TEMPORARY_;
② INFILE pop3 LENGTH=len missover;
  INPUT; * Open communication;
③ IF _INFILE_ =: '+OK' THEN DO;
     FILE pop3; PUT "USER myuserid";
     INPUT;
  END;
④ ELSE DO;
     ErrorMessage = 'No connection with server'; GOTO error;
  END;
  IF _INFILE_ =: '+OK' THEN DO;
     FILE pop3; PUT "PASS mypassword";
     INPUT;
  END;
  ELSE DO;
     ErrorMessage = 'User ID not accepted'; GOTO error;
  END;
  IF _INFILE_ =: '+OK' THEN DO;
     FILE pop3; PUT "STAT";
     INPUT;
  END;
  ELSE DO;
     ErrorMessage = 'Invalid UID or password'; GOTO error;
  END;
  IF _INFILE_ =: '+OK' THEN DO;
⑤    n_msg = INPUT(SCAN(_INFILE_,2),BEST.);
     IF n_msg GT 0 THEN DO;
        FILE pop3; PUT "LIST";
        INPUT;
     END;
```

Program 1 (continued)

```
      ELSE DO;
         ErrorMessage = 'No messages on server'; GOTO error;
      END;
   END;
   ELSE DO;
      ErrorMessage = 'Status not received'; GOTO error;
   END;
   IF _INFILE_ =: '+OK' THEN DO UNTIL (a=".");
⑥     INPUT @1 a $CHAR1. @;
      INPUT @1 msg_nr msg_length;
      IF a NE '.' THEN msglength{msg_nr}=msg_length;
   END;
⑦  DO n=1 TO n_msg;
      contents_start=0;
      tot_length=-5;
      FILE pop3; PUT "RETR " n;
      INPUT;
      IF _INFILE_ =: '+OK' THEN DO UNTIL (a="." and len=1);
         INPUT @1 a $CHAR1. @;
         INPUT @1 line $VARYING1000. len;
           TOT_LENGTH + len +2;
⑧        IF a='.' AND len GT 1 THEN DO;
            line=substr(line,2);
            tot_length +(-1);
         END;
         OUTPUT;
      END;
⑨     IF tot_length ge msglength{n} THEN DO;
         FILE pop3; PUT "DELE " n;
         INPUT;
      END;
      ELSE PUTLOG 'Message ' n 'not received correctly' ;
      IF _INFILE_ =: '+OK' THEN;
      ELSE PUTLOG 'Unable to delete message ' n;
   END;
   FILE pop3; PUT "QUIT";
   INPUT;
   STOP;
   RETURN;

⑩  error:
      PUTLOG errormessage;
      STOP;
   RETURN;
   RUN;
```

We need a two-way communication between SAS and the mail server. This means that the fileref of the FILENAME statement will be used both in INFILE and in FILE statements. The first reference should be in the INFILE statement ②. Together with the subsequent INPUT statement it opens the communication with the mail server. The mail server will respond with some greeting, starting with +OK and then normally the name of the server.

Note the use of the MISSOVER option on the INFILE statement. Since we are working with variable length records it is easy to look for some information (e.g. the server acknowledgement "+OK") on a line which is too short. MISSOVER prevents that SAS will look in the next line to find the required information

After successfully opening the communication (test for '+OK' ③) you have to send the user ID and password information to the server. If the connection could not be established write an error message to the log file and stop the processing ④.

If the logon has been established successfully we issue the STAT command. This will return information about the presence of mails. The second word in the response is the number of messages waiting to be retrieved. This is extracted from the response ⑤. If there are messages, we issue the LIST command, which will return a line for each message, containing the message number and it's size. The server will transmit a termination line, containing a single period, after all messages are reported. So we can use that to test whether we reached the end ⑥. The message length is stored in an array, to enable verification that we received the complete message. The array in the program can store information about 100 messages. That can be increased as needed.

Next we start a DO loop ⑦ to retrieve all messages one by one, using the RETR command. Just as with the response to the LIST command, the message is terminated with a line with a single period. To avoid problems with messages that contain a period on the first position of a line, an extra period is included. So this must be stripped off ⑧. Once the termination line is reached, we test whether we have received the complete message, by comparing the calculated byte count with the stored byte count in the array ⑨. If it is OK, the message is deleted from the server.

When all messages are retrieved we end the communication with the server, using the QUIT command, after which the DATA-step stops.

If any of the steps in the dialog resulted in an error response, we create an error message and branch to the error routine at the end of the program ⑩, which simply sends a message to the log and stops the DATA step.

The program reads all header information and stores that with the rest of the message. If you are not interested in all the header information you could filter the relevant lines by adding a filter routine between the points ⑧ and ⑨. That is left to the reader.

The program does not cater explicitly for mails with attachments. In case an attachment is present there are header lines that report that the message has multiple parts. These are sent in a continuous stream, with separator lines. It may take some trial and error, but they can be separated!

TEST RUN

For testing I inserted a PUTLOG “location:” _INFILE_; statement after each INPUT statement, where “location” is a label to recognize after which INPUT the PUTLOG was issued. Thus we can see what is happening in the SAS Log (these statements are not included in Program 1).
I sent two messages to my email account from two different senders and retrieved them with the program.

Listing 1 contains some relevant parts of the SAS log after running the program. I left out most of the header lines and almost everything of the second message as it would become rather boring…

Right after the general information regarding the connection information you recognize the sequence of the log on commands, followed by the STAT command to know whether there are messages. The LIST command is followed by two message lines providing the length of each message.

Then starts the section to retrieve the message. First all header lines and finally the body. After reading the termination line the DELE command and the start of the next message. After the last message you see the log-off response.

PROTOCOL ANALYSIS

While testing programs based on the SOCKET access method it is very handy to have a program that can display exactly what is sent over the TCP/IP line in either direction. My favorite is Ethereal, which can be downloaded from www.ethereal.com. It has a lot of options, but frankly I hardly use them. I just start the logging, send a message or something similar, stop the logging and analyze.

Listing 1: Partial SAS log of the test run

```
NOTE: The file/infile POP3 is:
      Local Host Name=notebook,
      Local Host IP addr=127.0.0.123.129,
      Peer Hostname Name=pop.wxs.nl,
      Peer IP addr=213.75.3.24,Peer Name=N/A,
      Peer Portno=110,Lrecl=256,Recfm=Variable

open communication: +OK kpnxchange.com
after sending user ID: +OK
after sending password: +OK User successfully logged on.
after STAT command: +OK User successfully logged on.
after LIST command: +OK 2 1728
Scanning end of input: 1 819
Message line: 1 819
Scanning end of input: 2 909
Message line: 2 909
Scanning end of input: .
Message line: .
after retrieve: +OK
Scanning end of message: Received: from cpsmtp-eml105.kpnxchange.com ([10.94.168.105]) by CPEXBE-
EML16.kpnsp.local with Microsoft SMTPSVC(6.0.3790.3959);
Getting message line: Received: from cpsmtp-eml105.kpnxchange.com ([10.94.168.105]) by CPEXBE-
EML16.kpnsp.local with Microsoft SMTPSVC(6.0.3790.3959);
Scanning end of message:       Wed, 26 Nov 2008 22:31:04 +0100
Getting message line:   Wed, 26 Nov 2008 22:31:04 +0100

(...)

Scanning end of message: From: Synchrona <synchrona@planet.nl>
Getting message line: From: Synchrona <synchrona@planet.nl>

(...)

Subject: Test message 1: Scanning end of message
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Getting message line

(...)

Getting message line: Read it in the DATA step
Scanning end of message

(...)

Scanning end of message: .
Getting message line: .
After DELE command: +OK
After retrieve: +OK

(...)

Scanning end of message: .
Getting message line: .
After DELE command: +OK
After QUIT command: +OK kpnxchange.com
```

Listing 1 (continued)

```
NOTE: 54 records were read from the infile POP3.
      The minimum record length was 0.
      The maximum record length was 128.
NOTE: 9 records were written to the file POP3.
      The minimum record length was 4.
      The maximum record length was 13.
NOTE: The data set WORK.MAILFILE has 41 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           0.39 seconds
      cpu time            0.20 seconds
```

HOW TO RUN THIS PROGRAM REPEATEDLY?

The program just accessed the mail server once to retrieve any messages and stopped. How can we achieve that such program interrogates the mail server repeatedly, e.g. every 10 minutes? There are two fundamentally different approaches to this. The first is that many operating systems provide mechanisms, which enable the time initiated dispatch of programs. In Windows this mechanism is called "scheduled tasks". The second approach is to implement it completely within SAS. An important difference between these two solutions is that using the operating system means that you start a SAS session when it is time to execute the program, while in the SAS solution a SAS session is always present, but in a dormant state until it is time to run.

SOLUTION 1: USING THE OPERATING SYSTEM

A general discussion of how to create scheduled tasks in all available operating systems is beyond the scope of this paper. Let us just look at Windows. The menu's and screenshots are based on Windows XP/SP3.

In the Control Panel you will find a folder named "Scheduled Tasks". As you guessed this is where you can find all scheduled tasks. It is also the place where you create one.

In Program 1 we accessed the mail server only once and stored the received messages in a WORK data set. For the scheduled program we want to accumulate the incoming mails in a permanent data set. Let us write a little program for that purpose (Program 2). It just allocates the permanent data library, then includes the source of Program 1, runs the PROC APPEND to accumulate all mails over the runs. Let us assume that the program is saved as "MailClient.sas" in the folder "e:\SAS Forum\source\".

When opening the "Scheduled Tasks" folder, the first item is "Add Scheduled Task". Opening it starts the Scheduled Tasks wizard. (Figure 1). The first question is which program to run. Select SAS and click Next.

Program 2: Program for scheduled task

```
LINAME sasdata "e:\sas forum\sasdata";
%INCLUDE "e:\SAS Forum\source\POP3.sas";
PROC APPEND BASE=sasdata.mailfile DATA=mailfile;
RUN;
```

On the next screen (Figure 2) you can specify the name under which the program will be known in the Scheduled Tasks folder, for instance "SAS Incoming mail" and also how often you want to run it. Choose one, you can always change it later (and then you have more options to tailor it to your needs).

Clicking NEXT brings you to the schedule screen (Figure 3) where you specify at what times the program should run. Finally you find a user information screen (Figure 4) where you specify under which username the program should execute. The wizard is ending here, but you still have not specified which program SAS should run. So check "Open advanced properties..." before finishing (Figure 5).

This opens a new window (Figure 6), where you can set more options than the wizard allows you. Start in the Run box. The pre-built contents is something like:

C:\PROGRA~1\SAS\SAS9~1.1\sas.exe -CONFIG "C:\Program Files\SAS\SAS 9.1\nls\en\SASV9.CFG"

Add to this command the SYSIN parameter to point to the program that has to run, like:

C:\PROGRA~1\SAS\SAS9~1.1\sas.exe -CONFIG "C:\Program Files\SAS\SAS 9.1\nls\en\SASV9.CFG" –SYSIN "E:\SAS Forum\source\MailClient.sas"

You could also enter other options, e.g. a location where the SAS Log should be stored.

In the Schedule tab of the screen you can change the run schedule. It has an advanced option that provides more scheduling options, e.g. run every hour.

Once you have done all this, the program should start running at the specified moments.

Figure 1: Scheduled Task Wizard – screen 1

Figure 2: Scheduled Task Wizard – screen 2

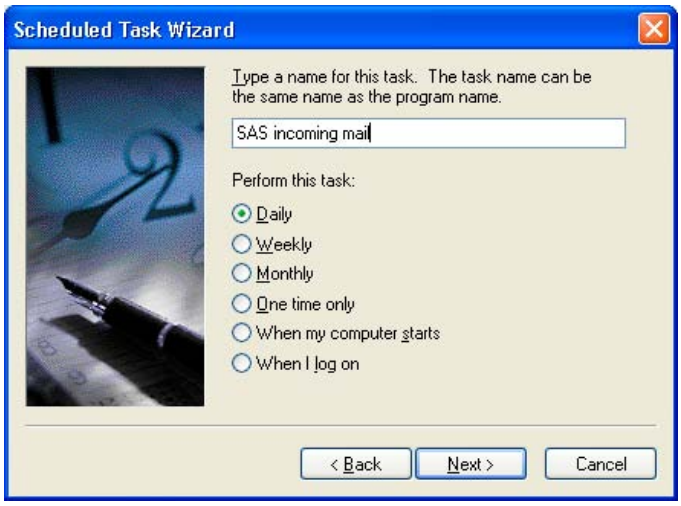
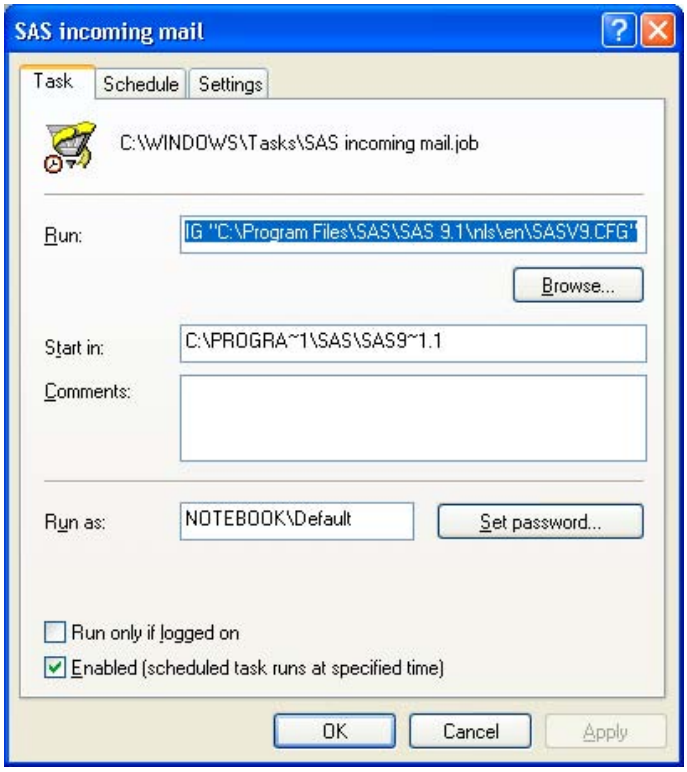


Figure 3: Scheduled Task Wizard – screen 3

Figure 4: Scheduled Task Wizard – screen 4

Figure 5: Scheduled Task Wizard – screen 5

Figure 6: Scheduled Task: more options and advanced scheduling



SOLUTION 2: REPEATED TASKS WITHIN SAS

The SLEEP function pauses the processing of a DATA step the specified time and then resumes processing. By wrapping it in the %SYSFUNC macro function ②, you can use it within a macro. That is the basis for this solution. We create a macro that runs "forever" (Program 3), at least as long as macro variable Quit is not equal to1. ①

Each iteration of the DO-loop in the macro starts with the sleep period. In the example it is 30 seconds ②. Then it fires off another macro, called %getmail, which is just Program 2 with a %MACRO statement in front and a %MEND statement at the end. It will be obvious that the counter and the %PUT statements are only there to send some information to the log, but are not relevant for the result.

How do you change Quit to 1 while the macro is running? That is accomplished via the %INCLUDE of the file "setexit.sas" ③. This file contains just one line: %LET Quit = 0; When you open that file outside SAS, e.g. with Notepad and change the statement into %LET Quit = 1; while the macro execution is in it's sleep phase then the next %INCLUDE sets Quit to 1 and the loop is terminated and the macro ends.

Program 3: Driver to run a program repeatedly

```
   %macro timer;
   %let quit=0;
   %let counter=0;
①  %do %until (&quit = 1);
      %let counter = %eval(&counter +1);
      %put sleep start &counter;
②     %let x = %sysfunc(sleep(30,1));
      %put sleep end;
      %getmail;
③     %include 'e:\SAS Forum\setexit.sas';
   %end;
   %mend;
   %timer
```

The sleep period in the SLEEP function is in seconds. It is specified using two arguments. The first argument specifies the quantity and the second is a multiplication factor. Sleep(30,1) sleeps for 30 seconds. Sleep(15,0.1) sleeps for 1.5 second. Sleep(0.001,3600) sleeps for 3.6 seconds. The default multiplication in the Windows environment is 1 (one second) and in other system environments 0.001 (milliseconds). The maximum period is 47 days.

When you are just busy updating the setexit.sas file at the moment that the %INCLUDE is executed, SAS will issue and error indicating that it cannot include the file and continue with the macro. Once you have saved and closed the file, the macro will pick up the new value during its next round.

CONCLUSION

Using the TCP/IP features of the FILENAME statement it is easy to develop a POP3 mail client in a DATA step. Once you have collected the incoming mails in a SAS data set you have the full power of SAS for further processing. Running a SAS program repeatedly can be accomplished using operating system facilities like the Window's Scheduled Tasks, but it also possible to write a simple driver macro that takes care of it inside SAS.

RECOMMENDED READING

More on the POP3 protocol can be found in the document on the web site www.faqs.org/rfcs/. Look for the pages RFC 1939, RFC 2449 and RFC 2595.

A general description of several of the FILENAME options can be found in the paper:
Michael Davis: *Reading From Alternate Sources: What To Do When The Input Is Not a Flat File*, SUGI 27 Proceedings p-006-27

The SAS 9.2 (9.1) Language Reference: Dictionary contains the formal description of the FILENAME – SOCKET statement.

CONTACT INFORMATION

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

Name	Erik Tilanus
Address	Horstlaan 51
Postal code, City	3971 LB Driebergen
Country	the Netherlands
Work Phone:	+31 343 517007
Fax:	+31 343 517007
E-mail:	erik.tilanus@planet.nl
Web:	www.synchrona.nl

At the website you can also find other presentations by the author, held at previous SUGI and SAS Forum meetings.