Receipt Filter

A tool for keeping an eye on read receipts that Mail.app automatically generates.

This code falls under the same license agreement as the MiscKit, you can do anything you want with this code. If you use it in your own program, however, you must acknowledge it's use in an app's Help or Info panel and/or in a README file accompanying it. (Just give credit where credit is due!)

Installation:

1) Copy the binary to /usr/local/lib/receiptfilter (or wherever you want it) or see "Compiling the Source Yourself" below. Yes, read it!
2) Start Mail.app.
3) In Mail.app's Preferences panel, choose "Expert" and change the field labeled "Mailer:" to "/usr/local/lib/receiptfilter".

4) Create a file called .rfrc in your home directory according to the directions below. Even if the file is left empty, it must be created. If you forget, the first time you run the program, an alert panel will pop up to tell you that the file couldn't be opened, that it is now being created.

How it works:

Receiptfilter is now the mailer for Mail.app, meaning that all outgoing mail is handed to this program for delivery. Mail.app executes receiptfilter with the recipient of the read receipt as its first argument and with the message on standard input. The program copies the message from stdin to a temporary file in order to scan it. The program first checks the header of the outgoing message to find the line:

Subject: Read Receipt

If this string is not found, the program gives the message to the mailer specified in the .rfrc file. If the message is a read receipt, it checks the strings specified in the user's .rfrc file to see if they are substrings of the message destination. If no matches are found, an alert panel pops up giving the user the choice of passing the receipt on to the mailer or

exiting the program without sending the receipt. If the line matches, the message will be passed on to the mailer, or not sent at all depending on whether the matching line begins with the character "P" or "F" respectively. When receiptfilter passes the message on to the real mailer, it makes the temporary file stdin once again and calls the real mailer with the original argument list.

Format of the ~/.rfrc file:

Each line in the configuration file begins with a single letter which indicates what the remainder of the line (minus leading and trailing whitespace) means:

M is followed by the absolute path for the mailer to which receiptfilter should pass messages. /usr/lib/sendmail is used if this line isn't specified.

L is followed by an absolute path or ~/path in user's home directory to the desired log file. This turns on verbose logging which records whether a read receipt was passed or filtered, the recipient, and the receipt body.

l is followed by an absolute path or ~/path in user's home directory to the desired log file. This turns on verbose logging which records whether a read receipt was passed or filtered, and the recipient. (This is a lower case ell).

A the rest of the line is not used. When the alert panel pops up because an address occurs which is not matched in the configuration file, this sets the default state of the switch box to on. This determines whether or not to append the new address to the configuration file with the pass/filter option selected by the user. Do not add this line if you want the default state to be off.

P is followed by a regular expression or string. If this pattern matches a substring of the read receipt's destination address, the receipt will be sent.

F is followed by a regular expression or string. If this pattern matches a substring of the read receipt's destination address, the receipt will **not** be sent.

A basic .rfrc file:

```
M    /usr/local/lib/mymailer
L    ~/.rflog
A
P    pmarc
F    byu.edu
```

This line filters all read receipts:

F .*

Important Note: If multiple matches are found for the same type of line (M, P, L, I, F), the last line matched will be the one used. I also recommend a review of regular expression syntax. If none of this makes sense, just create an empty .rfrc file for now and play around with the program. My next project is to create an app which will allow even the Unix novice to configure receiptfilter correctly.

Compiling the source yourself:

If you wish to compile the source yourself, you will first need to install the MiscKit since this program uses the MiscString class extensively. A current version of the MiscKit can be obtained from ftp.byu.edu somewhere under /pub/next. The MiscKit is a collection of general purpose objects and palettes which can be used in PD, shareware and commercial software as long as its use is acknowledged. The original author of receiptfilter, Steve Hayman, has made contributions to this kit and the kit is maintained by Don Yacktman. The kit is at least worth looking over and much of it is quite useful. If

you would like more information or want to be added to the MiscKit mailing list, e-mail don@darth.byu.edu. In the future it may be possible to obtain only the portion of the MiscKit library necessary to compile this program (You could always do it yourself from the MiscKit source for the MiscString class). There is currently discussion going on about coming up with reasonable divisions for the MiscKit libraries.

Fixes and enhancements in 1.2:

The real mailer is now specified in the .rfrc file. This way, the source does not have to be recompiled if the mailer on the system is changed. This also allows the binary to be used by those who don't have a compiler or don't wish to obtain the MiscKit.

RFC 821 is the SMTP definition which requires each line of text in a message to be no longer than 990 characters including the end of line character. This does not really detail a fix, it just is a reassurance that having linebuf be 1024 characters long is sufficient. If anybody can think of a standard e-mail implementation which would be broken by a 1024 character line length, let me know.

The program now only checks the header of the message for the "Subject: Read

Receipt" line. If I remember correctly, the standard states that the header and body of a message are to be separated by a single blank line. The program stops looking once it encounters this line.

It is now possible to specify e-mail addresses to which messages should be automatically passed or filtered.

An option for adding unmatched e-mail to the configuration file.

An option for logging read receipts and what was done with them.

All of this stuff was removed from the source code and placed in this readme file. A big :-) for Steve. I did try to comment the code as well as he originally did though. Actually, it has been over-commented so that somebody new to Obj-C and NeXTSTEP programming can get a better understanding of how it works. It is intended as both an example and a program which many people find useful.

I didn't change the installation procedure. Changing the Mailer default upon installation really only benefits whoever installs this, and if it's installed by root...

This will be moot when the app part of this program is done.

Comments and suggestions are welcome to both of us.
Send v1.2 bug reports to Paul.

Steve Hayman
shayman@Objectario.com

Paul M. Cardon
pmarc@zapotec.math.byu.edu