Email Reduction through SpontaneousRecall, Vacation & Expiry

Version 1.1
Release date: 2008-06-26

by Tom Nolf

University of Innsbruck
thomas.nolf@student.uibk.ac.at

1 Introduction

You just downloaded a Java package that was developed to give people the possibility to reduce the amount of emails by recalling messages or by providing an expiry date. Although there is only one jar-file there are three independently working applications, namely **Vacation**, **SpontaneousRecall** and **Expiry**.

1.1 Vacation

at.nolf.Vacation was designed to run directly on a mailserver. As its name suggests, this program is capable of responding to emails with vacation messages. *Vacation* includes a random key in the vacation message so that people can simply reply to it in order to recall their initial email. *Vacation* uses an email template to create vacation messages and does only respond with a vacation message to the same sender within a configurable time interval.

1.2 SpontaneousRecall

at.nolf.SpontaneousRecall enables users to recall any email as long as it hasn't already been fetched. People can resend a sent email marking it with a special label in the subject, which is then called a recall email. When receiving such a recall email, *SpontaneousRecall* tries to find the initial appropriate email by applying various algorithms. Once it has found exactly one match the email gets either deleted or marked as recalled to be filtered by rules of various email clients. The behaviour is configurable.
SpontaneousRecall can operate in three different ways. It can be instantiated and run directly on the mailserver, started manually on a remote machine to run once, or run as a daemon on a remote machine in a configurable time interval.

1.3 Expiry

at.nolf.Expiry is capable of finding and deleting emails in your mailbox that are expired. People can provide expiry dates in a headerfield or in the subject of

their emails that define the duration of validity. When this duration is expired the email gets deleted or marked as expired by this program, which can be run directly on the mailserver or on a remote machine.

Application	Utilization	Runs on
Vacation	Recall emails by replying to a vacation message	Mailserver
SpontaneousRecall	Recall any email	Mailserver or Remote Machine
Expiry	Deletes expired emails	Mailserver or Remote Machine

Table 1. Overview of Utilization

2 Installing, Configuring and Using it

2.1 Vacation

Example Sequence for Understanding
After someone sends you an email (s)he gets an automatic vacation message where you may suggest the possibility to recall his email. In order to recall the initial email the person simply has to reply to the vacation email. If (s)he additionally wants to get an acknowledgement, (s)he has to insert the label *ACK* at the very beginning of the subject when replying to the vacation email.

Requirements

- Write access to mailserver by ftp, scp or similar to upload the files
- Installed JRE 5 or higher on the mailserver
- Installed and running procmail

Installation
Installation is very easy. Just upload the jar-file into the root directory of your mailserver and edit .procmailrc with your favorite editor (ie. vim) to bring it to instantiate *Vacation* upon arrival of new email. This can be achieved by copying the following snippet in your .procmailrc before the part where the email gets delivered to INBOX:

```
# comment those 3 lines if you are not on vacation
:0ic
```

```
*
| java -cp EmailReduction.jar at.nolf.Vacation

:0B
* .*\<RECALL-1\>thomas.nolf@student.uibk.ac.at\</RECALL-1\>
| java -cp EmailReduction.jar at.nolf.Vacation recall
```

The first three lines after the comment call *Vacation* upon each arrival of email so that it can send a vacation message if necessary. You should comment those 3 lines with the character '#' if you are not on vacation.

The second block is responsible for recognizing a reply to a vacation message and starting *Vacation* in recall mode. You should put your email address between the tags, which you specified in the configuration file connection.properties (key: mail.smtp.from, see chapter 2.8 for details).

Template for Vacation message
Since *Vacation* uses a template to create a vacation message, one can create and modify it ad libitum. The template must be a text file named vacation.msg which has to be located in the same directory as the jar-file. Here is the content of my vacation.msg:

```
From: Thomas Nolf <thomas.nolf@student.uibk.ac.at>
Subject: Urlaub/vacation (Re: $SUBJECT)
Precedence: bulk

I am on vacation until xxxx.
Please refer all urgent business to xxx@uibk.ac.at
Tom Nolf

########################################################################
# You can recall your message (delete it from my mailbox)
# as long as I haven't read it.
# To do so, simply reply to this message without altering the body.
# $RECALL-LINE-1
# $RECALL-LINE-2
########################################################################
```

Explanation: It is important to know that all lines until the first empty line are interpreted as header fields for the vacation message. You can add any header field according to the specification (RFC 2822).

$SUBJECT and the lines $RECALL-LINE-X are placeholders that are substituted by the application. $SUBJECT is replaced by the subject of the initial email, $RECALL-LINE-1 is substituted with the generated random key

that makes it possible to delete the initial email by simply replying to it and $RECALL-LINE-2 will contain your email address to uniquely assign the message. Both $RECALL-LINE-1 and $RECALL-LINE-2 are mandatory in order to allow recalling of emails.

Template for Confirmation messages
Vacation (and also *SpontaneousRecall*) uses templates to create the confirmation messages about the successful or unsuccessful recall. Both templates must be text files in the same format as vacation.msg and named ack_positive.msg and ack_negative.msg respectively. One can use the placeholder $SUBJECT in these templates which will be replaced with the subject of the recall email. Here is the content of my ack_negative.msg:

```
Precedence: bulk
X-No-Archive: yes
Subject: Recall of $SUBJECT was not successful

Your recall of the message with the subject $SUBJECT was not
successful. Most likely the email has been fetched already.
```

Configuration
Vacation needs to know a few things about your mailaccount to work accordingly and expects this information in the file connection.properties. This file can easily be created using the tool *at.nolf.Configuration* or manually. See chapter 2.6 about how to create a configuration with this tool and chapter 2.8 to see all possible configurations.

Logs and Statistics
Detailed logs of *Vacation* are written to the file recall.log, but it also writes less detailed statistical information to a separate file called vacation_statistics.log to provide a better overview of the number of successful and unsuccessful recalls.

See chapter 2.7 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.2 SpontaneousRecall

Example Sequence for Understanding
Imagine someone sends you an email and wants that to be undone as long as you haven't fetched it. In this case the person has to forward the sent message without altering its content and indicate it as a recall email by specifying a special recall label (i.e. *RECALL-ME*) at the very beginning of the subject. If (s)he additionally wants to get a confirmation of the recall, (s)he has to append the label *-ACK* to the defined recall label when forwarding the email.

Example:

```
Subject of initial email: Hello World
Subject of recall email: RECALL-ME Hello World
Subject of recall email with confirmation: RECALL-ME-ACK Hello World
```

General Configuration
SpontaneousRecall needs to know a few things about the mailaccount that it should handle. It shares this information with *Vacation* and therefore reads all needed configurations from the file connection.properties. Depending on whether *SpontaneousRecall* runs directly on the mailserver or on a remote machine, the hostnames may slightly differ (ie. localhost, mail2.uibk.ac.at).

Moreover the behaviour of *SpontaneousRecall* needs to be configured by providing a file named recall.properties. There you can define if a recalled email gets deleted or just marked, the label that should be specified in the subject to recall an email, and the algorithms that should be applied to find the initial email to recall.

Both configuration files can easily be created by using the tool *at.nolf.Configuration* or even manually. See chapter 2.6 about how to create a configuration with this tool and chapter 2.8 to see all possible configurations.

Templates for Confirmation Messages
See chapter 2.1 (Templates for confirmation messages) for details.

Logs and Statistics
Detailed logs of *SpontaneousRecall* are written to the file recall.log, but it also writes less detailed statistical informations to a separate file recall_statistics.log to provide a better overview of the number of successful and unsuccessful recalls.

See chapter 2.7 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.3 SpontaneousRecall on Mailserver

Requirements

- Write access to mailserver by ftp, scp or similar to upload the files
- Installed JRE 5 or higher on the mailserver
- Installed and running procmail

Installation
Installation is very easy. If you didn't already upload the files for *Vacation*, do it now. Afterwards edit .procmailrc with your favorite editor (ie. vim) to bring it to instantiate *SpontaneousRecall* on arrival of email. This can be achieved by copying the following snippet to your .procmailrc **after** the part where the email gets delivered to INBOX. It's very important that *SpontaneousRecall* is run *after* the email was delivered to your mailbox, because it searches and deletes the email directly from your mailbox and therefore the email has to be in your INBOX before running.

```
:0wc
| $DELIVER +INBOX

:0Wi
* ^Subject:.*RECALL-ME.*
| java -cp EmailReduction.jar at.nolf.SpontaneousRecall

# E
:0
|
```

The first block delivers the email to your INBOX. Pay attention to append the character 'c' also to your recipe so that the incoming email gets cloned before it gets delivered to your mailbox so that the processing of the procmailrc-file is continued.

The second block calls *SpontaneousRecall* to recall the email only if the recall email contains the label RECALL-ME in the subject. Be sure to use the same label that you configured in the file recall.properties.

The last block deletes a remaining email if it wasn't a recall email. This is necessary since procmail expects all emails to be worked up after processing .procmailrc. Since we cloned the email for delivering we may still have one email left if it wasn't handled in the block before.

2.4 SpontaneousRecall on a Remote Machine

Requirements

- Write access to an arbitrary directory
- Installed JRE 5 or higher

Installation and Configuration
Installation is done like on the mailserver. Just copy the files to an arbitrary directory on your machine. Run the tool *at.nolf.Configuration* or adjust the properties connection.properties and recall.properties by hand. See chapter 2.6 and 2.8 for details about the tool and the possibilities of configuration.

SpontaneousRecall can be started to run once by executing the following command in your command line:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall
```

Or you may start it to run as a daemon by executing:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall daemon
```

Note that the time interval of the daemon is configured in the file recall.properties.

2.5 Expiry

As mentioned in chapter 1.3, *Expiry* is a program that finds and deletes expired emails. People can provide an expiration date of two different types in their emails that is supported by this program:

- Expiry date in a headerfield:
  The name of the headerfield is *'Expires'* and the value must follow the date format of RFC 2822. An Example for such a headerfield would be: "Expires: Fri, 29 Aug 2008 00:00:00 +0200"

- Expiry date in the subject:
  The expiry date must be given in the following format: "Expires: yyyy/mm/dd" whereby the position in the subject doesn't matter. For example: "Expires: 2008/09/28"

General Configuration
Like the two other programs, *Expiry* needs to know a few things about the mailaccount that it should handle. It shares this information with them and therefore reads all needed configurations from the file connection.properties. Depending on whether *Expiry* runs directly on the mailserver or on a remote machine, the hostnames may slightly differ (ie. localhost, mail2.uibk.ac.at).

Moreover the behaviour of *Expiry* needs to be configured by providing a file named expiry.properties, where you can define — among other things — if an expired email gets deleted or just marked. Such a file with default values will be created when you run this program for the very first time. See chapter 2.8 to see all possible configurations which you can manually change or add.

Requirements

- Write access to an arbitrary directory
- Installed JRE 5 or higher

Running Expiry

Expiry can be run once, or run as a daemon in configurable time intervals by executing one of the following commands in your command line:

```
java -cp EmailReduction.jar at.nolf.Expiry
```

or as a daemon:

```
java -cp EmailReduction.jar at.nolf.Expiry daemon
```

Note that the time interval of the daemon is configured in the file expiry.properties.

Logs and Statistics

Detailed logs of *Expiry* are written to the file recall.log, but it also writes less detailed statistical informations to a separate file expiry_statistics.log to provide a better overview of the number of successful expirations.

See chapter 2.7 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.6 Configuration Tool

To ease the process of configuration I developed a little tool that helps you with this. It's called *at.nolf.Configure* and can be run by executing the following command in your command line:

```
java -cp EmailReduction.jar at.nolf.Configure
```

It asks you some simple questions and writes your answers into appropriate configuration files. If some configuration files already exist, *Configuration* tells you the current value so that you can keep the old value by simply hitting the enter key.

Sample question:

```
Authentication necessary? [ true | false ] (current: true):
Username: (current: csae7569):
```

2.7 Changing the Log Configuration

One can modify the loglevel, maximum size of logfiles, rotations, etc. by providing one's own log4j configuration. Just unzip the file EmailReduction.jar and copy the file log4j.properties to the root directory. Use the file as a template and change desired lines. The default log4j configuration is overwritten by your own configuration by setting a system property containing the path to your file. Example:

```
java -Dlog4j.configuration=file:./mylog4j.properties \
          -cp EmailReduction.jar at.nolf.SpontaneousRecall
```

2.8 Complete Summary of Configuration

Table 2. Configuration with connection.properties (*Vacation*, *SpontaneousRecall* and *Expiry*)

Key	Description	Default
mail.pop3.host	Hostname of pop3 server	mail2.uibk.ac.at
mail.pop3.folder	Pop3 folder on server	INBOX
mail.pop3.auth	Is authentication at pop3 server necessary?	true
mail.transport.protocol	Protocol that is used to send an email	smtp
mail.smtp.host	Hostname of smtp server	localhost
mail.smtp.port	Port for smtp service	25
mail.smtp.from	Address of the sender (your email address)	You@uibk.ac.at
mail.smtp.starttls.enable	Allows communication over an encrypted layer	true
mail.smtp.auth	Is authentication at smtp server necessary?	false
session.debug	Log the process of sending an email very detailed	false
mail.user	Username for mailaccount (for both pop3 and smtp)	Your Username
mail.password	Encrypted password for mailaccount (pop3 and smtp)	
	Has to be set by using the tool *at.nolf.Configure*	Your Password
vacation.timespan	Time interval between sending two vacation messages	604800000
SendVacationMsgInMs	to one and the same recipient (in ms)	(=1 week)

Table 3. Configuration with recall.properties (only for *SpontaneousRecall*)

Key	Description	Default
recall.indicatorRecallMsg	Text that has to be specified at the very beginning of the subject to indicate a recall message	RECALL-ME
recall.markMessageAsRecalled	If true, the email to recall is only marked as recalled and not deleted	true
recall.markMessageAsRecalled.subject	This prefix is added to the subject to indicate that this email has been recalled	RECALLED:
recall.markMessageAsRecalled.header	This header is added to the recalled email	Recalled
recall.recallByMsgId	If true, *SpontaneousRecall* will try to find the initial email to recall by means of the message-id	true
recall.recallByTextCompare	If true, *SpontaneousRecall* will try to find the initial email to recall by comparing the content	true
recall.respectDateLastRun	If true, *SpontaneousRecall* will only search for recall messages that arrived since the last run	true
recall.timespanDaemonRun	Time interval between two runs of *SpontaneousRecall* when it operates in daemon mode (in ms)	30000

Table 4. Configuration with expiry.properties (only for *Expiry*)

Key	Description	Default
expiry.markMessageAsExpired	If true, the expired email is only marked as expired and not deleted	true
expiry.markMessageAsExpired.subject	This prefix is added to the subject to indicate that this email is expired	EXPIRED:
expiry.markMessageAsExpired.header	This header is added to the expired email	Expired
expiry.respectDateLastRun	If true, *Expiry* will only search for expired messages that arrived since the last run	true
expiry.timespanDaemonRun	Time interval between two runs of *Expiry* when it operates in daemon mode (in ms)	30000