Network Working Group S. Bosch
Internet-Draft February 10, 2013
Intended status: Standards Track
Expires: August 14, 2013

Sieve Email Filtering: Detecting Duplicate Deliveries
draft-bosch-sieve-duplicate-01

Abstract

This document defines a new test command "duplicate" for the "Sieve" email filtering language. This test adds the ability to detect duplicate message deliveries. The main application for this new test is handling duplicate deliveries commonly caused by mailing list subscriptions or redirected mail addresses. The detection is normally performed by matching the message ID to an internal list of message IDs from previously delivered messages. For more complex applications, the "duplicate" test can also use the content of a specific header or other parts of the message.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This is an extension to the Sieve filtering language defined by RFC 5228 [SIEVE]. It adds a test to determine whether a certain message was seen before by the delivery agent in an earlier execution of the Sieve script. This can be used to detect and handle duplicate message deliveries.

Duplicate deliveries are a common side-effect of being subscribed to a mailing list. For example, if a member of the list decides to reply to both the user and the mailing list itself, the user will get one copy of the message directly and another through mailing list. Also, if someone cross-posts over several mailing lists to which the user is subscribed, the user will receive a copy from each of those lists. In another scenario, the user has several redirected mail addresses all pointing to his main mail account. If one of the user's contacts sends the message to more than one of those addresses, the user will likely receive more than a single copy. Using the "duplicate" extension, users have the means to detect and handle such duplicates, e.g. by discarding them, marking them as "seen", or putting them in a special folder.

Duplicate messages are normally detected using the Message-ID header field, which is required to be unique for each message. However, the "duplicate" test is flexible enough to use different (weaker) criteria for defining what makes a message a duplicate, for example based on the subject line or parts of the message body. Other applications of this new test command are also possible, as long as the tracked value is a string.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEYWORDS].

Conventions for notations are as in [SIEVE] Section 1.1, including use of the "Usage:" label for the definition of action and tagged arguments syntax.

3. Test "duplicate"

```
Usage: "duplicate" [":handle" <handle: string>]
                   [":header" <header-name: string> /
                       ":value" <value: string>]
                   [":seconds" <timeout: number>]
```

In its basic form, the "duplicate" test keeps track of which messages were seen before by this test in an earlier execution of the Sieve script. Messages are identified by their message ID as contained in the Message-ID header. The "duplicate" test evaluates to "true" when the message was seen before and it evaluates to "false" when it was not.

As a side-effect, the "duplicate" test adds the message to an internal duplicate tracking list, so that the test will evaluate to "true" the next time the Sieve script is executed and the same message ID is encountered. Implementations MUST prevent adding messages to the internal duplicate tracking list when the Sieve script execution fails. If failed script executions would add messages to the duplicate tracking list, all "duplicate" tests in the Sieve script would erroneously yield "true" for the next delivery attempt of the same message, which can -- depending on the action taken for a duplicate -- easily lead to discarding the message without further notice.

For example, this can be implemented by deferring the definitive modification of the tracking list to the end of a successful Sieve script execution. This implementation is not without problems, however, as it can cause a race condition when a duplicate message is delivered in parallel before the tracking list is updated. This way, a duplicate message could be missed by the "duplicate" test. More complex implementations could use a locking mechanism to prevent this problem. But irrespective of what implementation is chosen, situations in which the "duplicate" test erroneously yields "true" MUST be prevented at all costs.

Implementations SHOULD limit the number messages that are tracked. Also, implementations SHOULD let entries in the tracking list expire after a short period of time. The user can explicitly control the length of this expiration time by means of the ":seconds" argument. If the ":seconds" argument is omitted, an appropriate default MUST be used. Sites SHOULD impose a maximum limit on the expiration time. If that limit is exceeded, the maximum value MUST silently be substituted; exceeding the limit MUST NOT produce an error.

By default, the tracked value is the content of the message's Message-ID header field. For more complex applications, the "duplicate" test can also be used to detect duplicate deliveries based on other message text. Then, the tracked value can be an arbitrary string value extracted from the message.

By adding the ":header" argument with a message header field name, the content of the specified header can be used as the tracked value instead of the default Message-ID header. Alternatively, the tracked

value can be specified explicitly using the ":value" argument. The string parameter of ":value" argument can be composed from arbitrary text extracted from the message using the "variables" [VARIABLES] extension. To extract text from the message body, the "foreverypart" and "extracttext" [SIEVE-MIME] extensions need to be used as well. This provides the user with detailed control over what identifies a message as a duplicate. The ":header" and ":value" arguments are mutually exclusive and specifying both for a single "duplicate" test command MUST trigger an error at compile time.

If the tracked value is extracted directly from a header, i.e. when the ":value" argument is not used, leading and trailing whitespace (see Section 2.2 of RFC 5228 [SIEVE]) MUST first be trimmed from the value before performing the actual duplicate verification. When the ":value" argument is used, such normalization concerns are the responsibility of the user.

The "duplicate" test MUST only check for duplicates amongst tracked values encountered in previous executions of the Sieve script; it MUST NOT consider tracked values encountered earlier in the current Sieve script execution as potential duplicates. This means that all "duplicate" tests in a Sieve scrip execution, including those located in scripts included using the "include" [INCLUDE] extension, MUST always yield the same result if the arguments are identical.

Using the ":handle" argument, the duplicate test can be employed for multiple independent purposes. The message is recognized as a duplicate only when the tracked value was seen before in an earlier script execution by a "duplicate" test with the same ":handle" argument.

NOTE: The necessary mechanism to track duplicate messages is very similar to the mechanism that is needed for tracking duplicate responses for the "vacation" [VACATION] action. One way to implement the necessary mechanism for the "duplicate" test is therefore to store a hash of the tracked value and, if provided, the ":handle" argument.

4. Sieve Capability Strings

A Sieve implementation that defines the "duplicate" test command will advertise the capability string "duplicate".

5. Examples

In the following basic example, message duplicates are detected by

tracking the Message-ID header. Duplicate deliveries are stored in a special folder contained in the user's Trash folder. If the folder does not exist, it is created automatically using the "mailbox" [MAILBOX] extension. This way, the user has a chance to recover messages when necessary. Messages that are not recognized as duplicates are stored in the user's inbox as normal.

```
require ["duplicate", "fileinto", "mailbox"];

if duplicate {
  fileinto :create "Trash/Duplicate";
}
```

The next example shows a more complex use of the "duplicate" test. The user gets network alerts from a set of remote automated monitoring systems. Multiple notifications can be received about the same event from different monitoring systems. The Message-ID of these messages is different, because these are all distinct messages from different senders. To avoid being notified multiple times about the same event the user writes the following script:

```
require ["duplicate", "variables", "imap4flags",
  "fileinto"];

if header :matches "subject" "ALERT: *" {
  if duplicate :seconds 60 :value "${1}" {
    setflag "\\seen";
  }
  fileinto "Alerts";
}
```

The subjects of the notification message are structured with a predictable pattern which includes a description of the event. In the script above the "duplicate" test is used to detect duplicate alert events. The message subject is matched against a pattern and the event description is extracted using the "variables" [VARIABLES] extension. If a message with that event in the subject was received before, but more than a minute ago, it is not detected as a duplicate due to the specified ":seconds" argument. In the the event of a duplicate, the message is marked as "seen" using the "imap4flags" [IMAP4FLAGS] extension. All alert messages are put into the "Alerts" mailbox irrespective of whether those messages are duplicates or not.

6. Security Considerations

A flood of unique messages could cause the list of tracked values to grow indefinitely. Implementations therefore SHOULD implement limits

on the number and lifespan of entries in that list.

7. IANA Considerations

The following template specifies the IANA registration of the Sieve extension specified in this document:

```
To: iana@iana.org
Subject: Registration of new Sieve extension

Capability name: duplicate
Description:     Adds test 'duplicate' that can be used to test
                 whether a particular message is a duplicate,
                 i.e. whether a copy of it was seen before by the
                 delivery agent that is executing the Sieve
                 script.
RFC number:      this RFC
Contact address: Sieve mailing list <sieve@ietf.org>
```

This information should be added to the list of sieve extensions given on http://www.iana.org/assignments/sieve-extensions.

8. References

8.1. Normative References

[INCLUDE] Daboo, C. and A. Stone, "Sieve Email Filtering: Include Extension", RFC 6609, May 2012.

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[SIEVE] Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language", RFC 5228, January 2008.

8.2. Informative References

[IMAP4FLAGS] Melnikov, A., "Sieve Email Filtering: Imap4flags Extension", RFC 5232, January 2008.

[MAILBOX] Melnikov, A., "The Sieve Mail-Filtering Language -- Extensions for Checking Mailbox Status and Accessing Mailbox Metadata", RFC 5490, March 2009.

[SIEVE-MIME]
Hansen, T. and C. Daboo, "Sieve Email Filtering: MIME Part Tests, Iteration, Extraction, Replacement, and Enclosure", RFC 5703, October 2009.

[VACATION]
Showalter, T. and N. Freed, "Sieve Email Filtering: Vacation Extension", RFC 5230, January 2008.

[VARIABLES]
Homme, K., "Sieve Email Filtering: Variables Extension", RFC 5229, January 2008.

Author's Address

Stephan Bosch
Enschede
NL

Email: stephan@rename-it.nl