Sieve Email Filtering: Detecting Duplicate Deliveries
draft-ietf-appsawg-sieve-duplicate-08

Abstract

   This document defines a new test command "duplicate" for the "Sieve"
   email filtering language. This test adds the ability to detect
   duplications. 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

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on December 28, 2014.

Copyright Notice

Table of Contents

1. Introduction

This document specifies an extension to the Sieve filtering language
defined by RFC 5228 [SIEVE]. It adds a test to track whether or not
a text string 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
often get one copy of the message directly and another through the
mailing list. Also, if someone cross-posts over several mailing
lists to which the user is subscribed, the user will likely 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 criteria for
defining what makes a message a duplicate, for example using the
subject line or parts of the message body. Other applications of
this new test command are also possible, as long as the tracked
unique 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> /
                       ":uniqueid" <value: string>]
                   [":seconds" <timeout: number>] [":last"]
```

The "duplicate" test identifies the message by a "unique ID", and, using that unique ID, keeps track of which messages were seen by a "duplicate" test during an earlier Sieve execution. In its basic form, the test gets the unique ID from the content of the message's 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 unique ID to an internal duplicate tracking list once the Sieve execution finishes successfully. The first time a particular unique ID is seen, the message is not a duplicate, and the unique ID is added to the tracking list. If a future Sieve execution sees a message whose unique ID appears in the tracking list, that test will evaluate to "true", and that message will be considered a duplicate.

Note that this side-effect is performed only when the "duplicate" test is actually evaluated. If the "duplicate" test is nested in a control structure or it is not the first item of an "allof" or "anyof" test list, its evaluation depends on the result of preceding tests, which may produce unexpected results

Implementations MUST only update the internal duplicate tracking list when the Sieve script execution finishes successfully. If failing script executions add the unique ID 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.

However, deferring the definitive modification of the tracking list to the end of a successful Sieve script execution is not without problems. 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.

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

The Messsage-ID header field is assumed to be globally unique as

required in Section 3.6.4 of RFC 5322 [IMAIL]. In practice, this
assumption may not aways prove to be true. The "duplicate" test does
not deal with this situation, which means that false duplicates may
be detected in this case. However, the user can address such
situations by specifying an alternative means of message
identification using the ":header" or the ":uniqueid" argument, as
described in the next section.

3.1. Arguments ":header" and ":uniqueid"

Duplicate tracking involves determining the unique ID for a given
message, and checking whether that unique ID is in the duplicate
tracking list. The unique ID for a message is determined as follows:

o When neither the ":header" argument nor the ":uniqueid" argument
   is used, the unique ID is the content of the message's Message-ID
   header field.

o When the ":header" argument is used, the unique ID is the content
   of the specified header field in the message. The header field
   name is not part of the resulting unique ID; it consists only of
   the field value.

o When the ":uniqueid" argument is used, the unique ID is the string
   parameter that is specified with the argument.

The ":header" and ":uniqueid" arguments are mutually exclusive and
specifying both for a single "duplicate" test command MUST trigger an
error.

The syntax rules for the header name parameter of the ":header"
argument are specified in Section 2.4.2.2 of RFC 5228 [SIEVE]. Note
that implementations MUST NOT trigger an error for an invalid header
name. Instead, the "duplicate" test MUST yield "false"
unconditionally in this case. The parameter of the ":uniqueid"
argument can be any string.

If the tracked unique ID value is extracted directly from a message
header field, i.e., when the ":uniqueid" argument is not used, the
following operations MUST be performed before the actual duplicate
verification:

o Unfold the header line as described in RFC 5322 [IMAIL], Section
   2.2.3. (see also Section 2.4.2.2 of RFC 5228 [SIEVE]).

o If possible, convert the header value to Unicode, encoded as UTF-8
   (see Section 2.7.2 of RFC 5228 [SIEVE]). If conversion is not
   possible, the value is left unchanged.

   o Trim leading and trailing whitespace from the header value (see
      [Section 2.2 of RFC 5228](#) [[SIEVE](#)]).

   Note that these rules also apply to the Message-ID header field used
   by the basic "duplicate" test without a ":header" or ":uniqueid"
   argument. When the ":uniqueid" argument is used, such normalization
   concerns are the responsibility of the user.

   If the header field specified using the ":header" argument exists
   multiple times in the message, extraction of the unique ID MUST use
   only the first occurrence. This is true whether or not multiple
   occurrences are allowed by [RFC 5322](#) [[IMAIL](#)], Section 3.6. If the
   specified header field is not present in the message, the "duplicate"
   test MUST yield "false" unconditionally. In that case the duplicate
   tracking list is left unmodified by this test, since no unique ID
   value is available. The same rules apply with respect to the
   Message-ID header field for the basic "duplicate" test without a
   ":header" or ":uniqueid" argument, since that header field could also
   be missing or occur multiple times.

   The string parameter of the ":uniqueid" 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 how
   the message's unique ID is created.

   The unique ID MUST be matched case-sensitively with the contents of
   the duplicate tracking list, irrespective of how the unique ID was
   determined. To achieve case-insensitive behavior when the
   ":uniqueid" argument is used, the "set" command added by the
   "variables" [[VARIABLES](#)] extension can be used to normalize the unique
   ID value to upper or lower case.

[3.2](#). Argument ":handle"

   The "duplicate" test MUST track a unique ID value independent of its
   source. This means that all values in the duplicate list should be
   used for duplicate testing, regardless of whether they were obtained
   from the Message-ID header, from an arbitrary header specified using
   the ":header" argument or explicitly from the ":uniqueid" argument.
   The following three examples are equivalent and match the same entry
   in the duplicate tracking list:

```
require "duplicate";
if duplicate {
  discard;
}
```

```
require "duplicate";
if duplicate :header "message-id" {
  discard;
}
```

```
require ["duplicate", "variables"];
if header :matches "message-id" "*" {
  if duplicate :uniqueid "${0}" {
    discard;
  }
}
```

The ":handle" argument can be used to override this default behavior.
The ":handle" argument separates a "duplicate" test from other
duplicate tests with a different or omitted ":handle" argument.
Using the ":handle" argument, unrelated "duplicate" tests can be
prevented from interfering with each other: a message is only
recognized as a duplicate when the tracked unique ID 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 unique ID and, if provided, the ":handle"
argument.

3.3. Arguments ":seconds" and ":last"

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, which
accepts an integer value specifying the timeout value in seconds. If
the ":seconds" argument is omitted, an appropriate default value MUST
be used. A default expiration time of around 7 days is usually
appropriate. Sites SHOULD impose a maximum limit on the expiration
time. If that limit is exceeded by the ":seconds" argument, the
maximum value MUST silently be substituted; exceeding the limit MUST
NOT produce an error. If the ":seconds" argument is zero, the
"duplicate" test MUST yield "false" unconditionally.

When the ":last" argument is omitted, the expiration time for entries
in the duplicate tracking list MUST be measured relative to the
moment at which the entry was first created; i.e., at the end of the
successful script execution during which "duplicate" test returned
"false" for a message with that particular unique ID value. This

means that subsequent duplicate messages have no influence on the
time at which the entry in the duplicate tracking list finally
expires.

In contrast, when the ":last" argument is specified, the expiration
time MUST be measured relative to the last script execution during
which the "duplicate" test was used to check the entry's unique ID
value. This effectively means that the entry in the duplicate
tracking list will not expire while duplicate messages with the
corresponding unique ID keep being delivered within intervals smaller
than the expiration time.

It is possible to write Sieve scripts where during a single execution
more than one "duplicate" test is evaluated with the same unique ID
value and ":handle" argument but different ":seconds" or ":last"
arguments. The resulting behavior is left undefined by this
specification, so such constructs should be avoided. Implementations
MAY choose to use the ":seconds" and ":last" arguments from the
"duplicate" test that was evaluated last.

3.4. Interaction with Other Sieve Extensions

The "duplicate" test does not support either the "index"
[DATE-INDEX], or "mime" [SIEVE-MIME] extensions directly, meaning
that none of the ":index", ":mime" or associated arguments are added
to the "duplicate" test when these extensions are active. The
":uniqueid" argument can be used in combination with the "variables"
[VARIABLES] extension to achieve the same result indirectly.

Normally, Sieve scripts are executed at final delivery. However,
with the "imapsieve" [IMAPSIEVE] extension, Sieve scripts are invoked
when the IMAP [IMAP] server performs operations on the message store,
e.g. when messages are uploaded, flagged, or moved to another
location. The "duplicate" test is devised for use at final delivery
and the semantics in the "imapsieve" context are left undefined.
Therefore, implementations SHOULD NOT allow the "duplicate" test to
be used in the context of "imapsieve".

4. Sieve Capability Strings

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

5. Examples

5.1. **Example 1**

   In this 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";
}
```

5.2. **Example 2**

   This example shows a more complex use of the "duplicate" test. The
   user gets network alerts from a set of remote automated monitoring
   systems. Several 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 more than a single time about the
   same event the user writes the following script:

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

if header :matches "subject" "ALERT: *" {
  if duplicate :seconds 60 :uniqueid "${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.

[5.3](#). **Example 3**

   This example shows how the "duplicate" test can be used to limit the
   frequency of notifications sent using the "enotify" [NOTIFY]
   extension. Consider the following scenario: a mail user receives
   XMPP notifications [NOTIFY-XMPP] about new mail through Sieve, but
   sometimes a single contact sends many messages in a short period of
   time. Now the user wants to prevent being notified of all of those
   messages. The user wants to be notified about messages from each
   person at most once per 30 minutes and writes the following script:

```
require ["variables", "envelope", "enotify", "duplicate"];

if envelope :matches "from" "*" { set "sender" "${1}"; }
if header :matches "subject" "*" { set "subject" "${1}"; }

if not duplicate :seconds 1800 :uniqueid "${sender}"
{
  notify :message "[SIEVE] ${sender}: ${subject}"
    "xmpp:user@im.example.com";
}
```

   The example shown above uses the message envelope sender rather than
   the Message-ID header as the unique ID for duplicate tracking.

   The example can be extended to allow more messages from the same
   sender in close succession as long as the discussed subject is
   different. This can be achieved as follows:

```
require ["variables", "envelope", "enotify", "duplicate"];

if envelope :matches "from" "*" { set "sender" "${1}"; }
if header :matches "subject" "*" { set "subject" "${1}"; }

# account for 'Re:' prefix
if string :comparator "i;ascii-casemap"
  :matches "${subject}" "Re:*"
{
  set "subject" "${1}";
}
if not duplicate :seconds 1800
  :uniqueid "${sender} ${subject}"
{
  notify :message "[SIEVE] ${sender}: ${subject}"
    "xmpp:user@im.example.com";
}
```

   This uses a combination of the message envelope sender and the

subject of the message as the unique ID for duplicate tracking.

5.4. Example 4

For this example, the mail user uses the "duplicate" test for two
separate applications: for discarding duplicate events from a
notification system and to mark certain follow-up messages in a
software support mailing as "seen" using the "imap4flags"
[IMAP4FLAGS] extension.

The two "duplicate" tests in the following example each use a
different header to identify messages. However, these "X-Event-ID"
and "X-Ticket-ID headers can have similar values in this case (e.g.
both based on a time stamp), meaning that one "duplicate" test can
erroneously detect duplicates based on ID values tracked by the
other. Therefore, the user wants to prevent the second "duplicate"
test from matching ID values tracked by the first "duplicate" test
and vice versa. This is achieved by specifying different ":handle"
arguments for these tests.

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

if duplicate :header "X-Event-ID" :handle "notifier" {
  discard;
}
if allof (
  duplicate :header "X-Ticket-ID" :handle "support",
  address "to" "support@example.com",
  header :contains "subject" "fileserver")
{
  setflag "\\seen";
}
```

6. Security Considerations

A flood of unique messages could cause the duplicate tracking list to
grow indefinitely. Therefore, implementations SHOULD limit the
number of entries in the duplicate tracking list. When limiting the
number of entries, implementations SHOULD discard the oldest ones
first.

Scripts using the duplicate test evaluation should be aware that
Message-IDs are not necessarily unique, either through the fault of
benign generators or attackers injecting a message with the
properties used by the duplicate Sieve filter at some point prior to
the Sieve filter. Therefore, scripts are well advised to be
conservative with respect to actions taken when duplicate messages

are identified only by Message-ID.

The list of unique IDs used for duplicate tracking can include
privacy-sensitive information, such as Message-ID values, content of
subject lines, and content extracted from message bodies.
Implementations SHOULD protect that information, by obscuring it
through hashing (see the note at the end of Section 3.2) and/or by
storing it with a level of access control equivalent to that of the
messages themselves.

These measures will not prevent an entity that has access to the
duplicate tracking list from querying whether messages with certain
unique ID values were received. As this operation is the essence of
the "duplicate" test, this cannot be prevented, and may violate the
expectations of the user. For example, a user who deletes a message
from the server may expect that no record of it remains on the
server, but that will not be true if its Message-ID is persisted on
the server in the duplicate tracking list.

It's notable, however, that server logs will often store the
information present on the duplicate tracking list anyway, and
probably would expose plaintext Message-IDs for a much longer period
than this mechanism would. Users of email services that
intentionally delete such logs with the intent of limiting
traceability should be made aware that use of the duplicate tracking
mechanism re-exposes this information for the duration of the expiry
interval. Therefore, a shorter default expiry interval may be
appropriate in those situations.

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. Acknowledgements

Thanks to Brian Carpenter, Cyrus Daboo, Arnt Gulbrandsen, Tony
Hansen, Kristin Hubner, Barry Leiba, Alexey Melnikov, Subramanian
Moonesamy, Tom Petch, Hector Santos, Robert Sparks, Aaron Stone, and
Stefan Winter for reviews and suggestions. With special thanks to
Ned Freed for his guidance and support.

9. References

9.1. Normative References

[DATE-INDEX]
            Freed, N., "Sieve Email Filtering: Date and Index
            Extensions", RFC 5260, July 2008.

[IMAIL]   Resnick, P., Ed., "Internet Message Format", RFC 5322,
            October 2008.

[IMAPSIEVE]
            Leiba, B., "Support for Internet Message Access Protocol
            (IMAP) Events in Sieve", RFC 6785, November 2012.

[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.

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

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

9.2. Informative References

[IMAP]   Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION
           4rev1", RFC 3501, March 2003.

[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.

[NOTIFY]   Melnikov, A., Leiba, B., Segmuller, W., and T. Martin,
           "Sieve Email Filtering: Extension for Notifications",
           RFC 5435, January 2009.

[NOTIFY-XMPP]
           Saint-Andre, P. and A. Melnikov, "Sieve Notification
           Mechanism: Extensible Messaging and Presence Protocol
           (XMPP)", RFC 5437, January 2009.

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

Author's Address

   Stephan Bosch
   Enschede
   NL

   Email: stephan@rename-it.nl