Network Working Group
Request for Comments: 5232
Category: Standards Track

A. Melnikov
Isode Limited

January 2008

# Sieve Email Filtering: Imap4flags Extension

## Status of This Memo



## Abstract

Recent discussions have shown that it is desirable to set different IMAP (RFC 3501) flags on message delivery. This can be done, for example, by a Sieve interpreter that works as a part of a Mail Delivery Agent.

This document describes an extension to the Sieve mail filtering language for setting IMAP flags. The extension allows setting of both IMAP system flags and IMAP keywords.

## Table of Contents

## 1. Introduction

This is an extension to the Sieve language defined by [SIEVE] for setting [IMAP] flags. It adds a new tagged argument to "keep" and "fileinto" that describes the list of flags that have to be set when the message is delivered to the specified mailbox. It also adds several actions to help manipulate list of flags and a test to check whether a flag belongs to a list.

From the user's perspective, this extension provides several capabilities. First, it allows manipulation of sets of IMAP flags. Second, it gives the ability to associate a set of IMAP flags with a message that is delivered to a mailstore using the keep/fileinto actions. Third, it maintains an internal variable. The internal variable contains the default flags that will be associated with a message that is delivered using the keep, implicit keep, or fileinto actions, when the :flags tagged argument is not specified. When the Sieve [VARIABLES] extension is also supported by the Sieve engine, it enables support for multiple variables containing sets of IMAP flags.

The capability string associated with the extension defined in this document is "imap4flags". All conformant implementations MUST implement all Sieve actions (setflag, addflag, removeflag), the "hasflag" test, and the ":flags" tagged argument described in this document.

The "imap4flags" extension can be used with or without the "variables" extension [VARIABLES]. When the "variables" extension is enabled in a script using <require "variables">, the script can use explicit variable names in setflag/addflag/removeflag actions and the hasflag test. See also Section 3 for more details. When the "variables" extension is not enabled, the explicit variable name parameter to setflag/addflag/removeflag/hasflag MUST NOT be used and MUST cause an error according to [SIEVE].

### 1.1. Conventions Used

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

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 RFC 2119.

## 2. General Requirements for Flag Handling

The following notes apply to processing of addflag/removeflag/setflag actions, the "hasflag" test and the ":flags" tagged argument.

A Sieve interpreter MUST ignore empty strings (i.e., "") in a list-of-flags parameter.

A string containing a space-separated list of flag names is equivalent to a string list consisting of the flags. This requirement is to simplify amalgamation of multiple flag lists.

The Sieve interpreter SHOULD check the list of flags for validity as described by [IMAP] ABNF. In particular, according to [IMAP], non-ASCII characters are not allowed in flag names. However, spaces MUST always be allowed as delimiters in strings representing a list of flags. In such strings, multiple spaces between flag names MUST be treated as a single space character, and leading and trailing spaces MUST be ignored.

If a flag validity check fails, the flag MUST be ignored.

Note that it is not possible to use this extension to set or clear the \Recent flag or any other special system flag that is not settable in [IMAP]. Any such flags MUST be ignored if included in a flag list.

## 3. Actions

All actions described in this specification (setflag, addflag, removeflag) operate on string variables that contain a set of [IMAP] flags. On variable substitution, a set of flags is represented as a string containing a space-separated list of flag names.

Any setflag/addflag/removeflag action MAY alter the flag list in any way that leaves its semantics as a set of case-insensitive words unaltered. For example, it may reorder the flags, alter the case of the letters in them, or add or remove duplicates or extraneous spaces. Scripts MUST NOT make assumptions about the ordering of flags in lists or the preservation of their case.

Note that the parameter specifying a variable name to setflag/addflag/removeflag actions and the hasflag test is optional. If the parameter is not specified, the actions operate on the internal variable, which has the empty value when the script starts execution. If the SIEVE interpreter doesn't support the "variables" extension [VARIABLES], the presence of the variable name parameter will cause a runtime error [SIEVE].

The "addflag" action adds flags to an existing set. The "removeflag" action removes flags from an existing set. The "setflag" action replaces an existing set of flags with a new set. The "set" action defined in [VARIABLES] can be used to replace an existing set of flags with a new set as well. However, it should be noted that the "set" action can't perform any flag reordering, duplicate elimination, etc.

The :flags tagged argument to "keep" and "fileinto" actions is used to associate a set of flags with the current message. If the :flags tagged argument is not specified with those two actions, the current value of the internal variable is used instead. The value of the internal variable also applies to the implicit keep.

Note that when keep/fileinto is used multiple times in a script and duplicate message elimination is performed (as described in Section 2.10.3 of [SIEVE]), the last flag list value MUST win.

## 3.1. Action setflag

```
Usage:   setflag [<variablename: string>]
         <list-of-flags: string-list>
```

Setflag is used for setting [IMAP] system flags or keywords. Setflag replaces any previously set flags.

```
Example:  if size :over 500K {
              setflag "\\Deleted";
          }
```

A more substantial example is the following:

```
Example:
     if header :contains "from" "boss@frobnitzm.example.edu" {
         setflag "flagvar" "\\Flagged";
         fileinto :flags "${flagvar}" "INBOX.From Boss";
     }
```

## 3.2. Action addflag

```
Usage:   addflag [<variablename: string>]
         <list-of-flags: string-list>
```

Addflag is used to add flags to a list of [IMAP] flags. It doesn't replace any previously set flags. This means that multiple occurrences of addflag are treated additively.

The following examples demonstrate requirements described in Section 2.1. The following two actions

```
addflag "flagvar" "\\Deleted";
addflag "flagvar" "\\Answered";
```

produce the same result as the single action

```
addflag "flagvar" ["\\Deleted", "\\Answered"];
```

or

```
addflag "flagvar" "\\Deleted \\Answered";
```

or

```
addflag "flagvar" "\\Answered \\Deleted";
```

### 3.3. Action removeflag

Usage: removeflag [<variablename: string>]
<list-of-flags: string-list>

Removeflag is used to remove flags from a list of [IMAP] flags. Removeflag clears flags previously set by "set"/"addflag". Calling removeflag with a flag that wasn't set before is not an error and is ignored. Note that if an implementation doesn't perform automatic duplicate elimination, it MUST remove all occurrences of the flags specified in the second parameter to removeflag. Empty strings in the list-of-flags MUST be ignored. Also note that flag names are case-insensitive, as described in [IMAP]. Multiple removeflag actions are treated additively.

Example:

```
if header :contains "Disposition-Notification-To"
   "mel@example.com" {
    addflag "flagvar" "$MDNRequired";
}
if header :contains "from" "imap@cac.washington.example.edu" {
    removeflag "flagvar" "$MDNRequired";
    fileinto :flags "${flagvar}" "INBOX.imap-list";
}
```

## 4. Test hasflag

Usage: hasflag [MATCH-TYPE] [COMPARATOR]
[<variable-list: string-list>]
<list-of-flags: string-list>

The hasflag test evaluates to true if any of the variables matches any flag name. The type of match defaults to ":is". If the list of variables is omitted, value of the internal variable is used instead.

The default comparator is "i;ascii-casemap", which is the same case-insensitive comparison as defined for IMAP flags by [IMAP].

The "relational" extension [RELATIONAL] adds a match type called ":count". The :count of a variable returns the number of distinct flags in it. The count of a list of variables is the sum of the counts of the member variables.

Example:
If the internal variable has the value "A B", the following test

```
hasflag :is "b A"
```

evaluates to true. The above test can also be written as

```
hasflag ["b","A"]
```

Example:
If the variable "MyVar" has value "NonJunk Junk gnus-forward $Forwarded NotJunk JunkRecorded $Junk $NotJunk", the following tests evaluate to true:

```
hasflag :contains "MyVar" "Junk"
hasflag :contains "MyVar" "forward"
hasflag :contains "MyVar" ["label", "forward"]
hasflag :contains "MyVar" ["junk", "forward"]
```

Note that the last of these tests can be rewritten as

```
hasflag :contains "MyVar" "junk forward"
```

or

```
hasflag :contains "MyVar" "forward junk"
```

However, the last two forms are not recommended.

And the following tests will evaluate to false:

```
hasflag :contains "MyVar" "label"

hasflag :contains "MyVar" ["label1", "label2"]
```

Example:
If the variable "MyFlags" has the value "A B", the following test

```
hasflag :count "ge" :comparator "i;ascii-numeric"
        "MyFlags" "2"
```

evaluates to true, as the variable contains 2 distinct flags.

## 5. Tagged Argument :flags

This specification adds a new optional tagged argument ":flags" that alters the behavior of actions "keep" and "fileinto".

The :flags tagged argument specifies that the flags provided in the subsequent argument should be set when fileinto/keep delivers the message to the target mailbox/user's main mailbox. If the :flags tagged argument is not specified, "keep" or "fileinto" will use the current value of the internal variable when delivering the message to the target mailbox.

Usage: ":flags" <list-of-flags: string-list>

The copy of the message filed into the mailbox will have only flags listed after the :flags tagged argument.

The Sieve interpreter MUST ignore all flags that it can't store permanently. This means that the interpreter MUST NOT treat failure to store any flag as a runtime failure to execute the Sieve script. For example, if the mailbox "INBOX.From Boss" can't store any flags, then

```
fileinto :flags "\\Deleted" "INBOX.From Boss";
```

and

```
fileinto "INBOX.From Boss";
```

are equivalent.

This document doesn't dictate how the Sieve interpreter will set the [IMAP] flags. In particular, the Sieve interpreter may work as an IMAP client or may have direct access to the mailstore.

## 6. Interaction with Other Sieve Actions

This extension works only on the message that is currently being processed by Sieve; it doesn't affect another message generated as a side effect of any action or any other message already in the mailstore.

The extension described in this document doesn't change the implicit keep (see Section 2.10.2 of [SIEVE]).

## 7. Security Considerations

Security considerations are discussed in [IMAP], [SIEVE], and [VARIABLES].

This extension intentionally doesn't allow setting [IMAP] flags on an arbitrary message in the [IMAP] message store.

It is believed that this extension doesn't introduce any additional security concerns.

## 8. IANA Considerations

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

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

Capability name: imap4flags
Description: Adds actions and tests for manipulating IMAP flags
RFC number: RFC 5232
Contact address: The Sieve discussion list <ietf-mta-filters@imc.org>

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

## 9. Extended Example

```
#
# Example Sieve Filter
# Declare any optional features or extension used by the script
#
require ["fileinto", "imap4flags", "variables"];

#
# Move large messages to a special mailbox
#
```

```
   if size :over 1M
           {
           addflag "MyFlags" "Big";
           if header :is "From" "boss@company.example.com"
                     {
   # The message will be marked as "\Flagged Big" when filed into
   # mailbox "Big messages"
                     addflag "MyFlags" "\\Flagged";
                     }
           fileinto :flags "${MyFlags}" "Big messages";
           }

   if header :is "From" "grandma@example.net"
           {
           addflag "MyFlags" ["\\Answered", "$MDNSent"];
   # If the message is bigger than 1Mb it will be marked as
   # "Big \Answered $MDNSent" when filed into mailbox "grandma".
   # If the message is shorter than 1Mb it will be marked as
   # "\Answered $MDNSent"
           fileinto :flags "${MyFlags}" "GrandMa";
           }

   #
   # Handle messages from known mailing lists
   # Move messages from IETF filter discussion list to filter folder
   #
   if header :is "Sender" "owner-ietf-mta-filters@example.org"
           {
           set "MyFlags" "\\Flagged $Work";
   # Message will have both "\Flagged" and $Work flags
           keep :flags "${MyFlags}";
           }

   #
   # Keep all messages to or from people in my company
   #
   elsif anyof address :domain :is ["From", "To"] "company.example.com"
           {
           keep :flags "${MyFlags}"; # keep in "Inbox" folder
           }

   # Try to catch unsolicited email.  If a message is not to me,
   # or it contains a subject known to be spam, file it away.
   #
   elsif anyof (not address :all :contains
                  ["To", "Cc"] "me@company.example.com",
                header :matches "subject"
                  ["*make*money*fast*", "*university*dipl*mas*"])
```

```
        {
        remove "MyFlags" "\\Flagged";
        fileinto :flags "${MyFlags}" "spam";
        }
else
        {
        # Move all other external mail to "personal"
        # folder.
        fileinto :flags "${MyFlags}" "personal";
        }
```

## 10. Acknowledgments

This document has been revised in part based on comments and discussions that took place on and off the Sieve mailing list.

The help of those who took the time to review the document and make suggestions is appreciated, especially that of Tim Showalter, Barry Leiba, Randall Gellens, Ken Murchison, Cyrus Daboo, Matthew Elvey, Jutta Degener, Ned Freed, Marc Mutz, Nigel Swinson, Kjetil Torgrim Homme, Mark E. Mallett, Dave Cridland, Arnt Gulbrandsen, Philip Guenther, Rob Austein, Sam Hartman, Eric Gray, and Cullen Jennings.

Special thanks to Tony Hansen and David Lamb for helping me better explain the concept.

## 11. Normative References

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

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

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

[RELATIONAL] Segmuller, W. and B. Leiba "Sieve Email Filtering: Relational Extension", RFC 5231, January 2008.

## Author's Address

Alexey Melnikov
Isode Limited

5 Castle Business Village
Hampton, Middlesex
United Kingdom, TW12 2BX

EMail: alexey.melnikov@isode.com

## Full Copyright Statement



## Intellectual Property