

Network Working Group
Internet Draft: Sieve -- IMAP flag Extension
Document: [draft-ietf-sieve-imapflags-04.txt](#)
Expires: August 2006

A. Melnikov
Isode Limited
February 2006

SIEVE Email Filtering: IMAP flag Extension

Status of this memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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".

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Abstract

Recent discussions have shown that it is desirable to set different [\[IMAP\]](#) 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 to set both [\[IMAP\]](#) system flags and [\[IMAP\]](#) keywords.

[@](#). Meta-information on this draft

This information is intended to facilitate discussion. It will be removed when this document leaves the Internet-Draft stage.

0.1. Discussion

This draft defines an extension to the Sieve mail filtering language ([RFC 3028](#)) being discussed on the MTA Filters mailing list at <ietf-mta-filters@imc.org>. Subscription requests can be sent to <ietf-mta-filters-request@imc.org> (send an email message with the word "subscribe" in the body). More information on the mailing list along with a WWW archive of back messages is available at <<http://www.imc.org/ietf-mta-filters/>>.

0.2. Changes from the version submitted to the Sieve mailing list

1. Added addflag and removeflag actions
2. Changed the semantics of setflag (setflag is not additive any more)
3. Corrected section "Interaction with Other Sieve Actions". Removed incorrect reference to the forward action as to an action that prohibits setflag.
4. Added paragraph about the mutual order of "fileinto"/"keep" and "setflag"/"addflag"/"removeflag" actions.

0.3. Changes from the revision 00

1. Corrected Capability Identifier section ([Section 2](#))
2. Corrected "Interaction with Other Sieve Actions" section ([Section 4](#))
3. Examples were updated to be compatible with Sieve-07 draft
4. Added "mark" and "unmark" actions

0.4. Changes from the revision 01

1. Some language fixes based on Tony Hansen comments
2. Clarified that the extension allows to set both IMAP System Flags and Keywords

0.5. Changes from the revision 02

1. BugFix: all backslashes must be escaped
2. Added extended example and more detailed description of

"addflag"/"removeflag" additivity.

3. Minor example bugfixes

0.6. Changes from the revision 03

1. Added second way to specify flags to be set (via optional tagged arguments). [Tim Showalter]
2. Rules for using Reject with imapflags relaxed. [Randall Gellens]
3. Removed ABNF section completely, added syntax description to action definition. [Tim Showalter]
4. Cleaned up the example. [Ken Murchison]
5. Added FM (Flag Manipulation) acronym.
6. Clarified "mark"/"unmark" behavior. [Randall Gellens]

0.7. Changes from the revision 04

1. "Interaction with Other Sieve Actions" was simplified based on comments from Tim Showalter. Added sentence saying that imapflags doesn't change an implicit keep.
2. Several editorial comments from Tim Showalter.

0.8. Changes from the revision 05

1. Updated copyright, author address, section numbers and references.
2. Added dependency on Sieve "variables" extension.
3. Several editorial comments from Matthew Elvey.
4. Removed "mark" and "unmark" actions.
5. Added "hasflag" test.
6. Dropped ":globalflags"
7. An invalid flag name doesn't cause a script execution failure anymore, as imapflags now depends on variables and a variable can have an arbitrary value.

0.9. Changes from the revision 06 ([draft-ietf-sieve-imapflags-00.txt](#))

1. Updated boilerplate and references.

2. Fixed capability string in the extended example.
3. Improved implementation using macros ([section 6](#)).

[0.10](#). Changes from [draft-ietf-sieve-imapflags-00.txt](#)

1. Added back the internal variable, made the variable parameter to all actions optional.
2. Some editorial suggestions from Mark E. Mallett.
3. Updated boilerplate once again.

[0.11](#). Changes from [draft-ietf-sieve-imapflags-01.txt](#)

1. Clarified that if "variables" is not available, then usage of the explicit variable parameter causes a runtime error.
2. Clarified handling of spaces, in particular that leading/trailing spaces in a flag variable are to be ignored.
3. Clarified that the extension can't be used to set the \Recent flag.
4. Made the variable list parameter to hasflag test optional, for consistency with all actions.
5. Dropped the "Implementation Notes" section.
6. Fixed an error in [section 5](#): when the :flags tagged argument is not specified, the correct behavior is to use flags from the internal variable.
7. Other minor editorial changes.

[0.12](#). Changes from [draft-ietf-sieve-imapflags-02.txt](#)

1. Changed "Syntax:" label to "Usage:".
2. Updated the Introduction section to mention that hasflag also has an optional parameter.
3. Clarified that a failure to store flags for a message is not a runtime error.
4. Minor editorial nits from Philip.
5. Addressed ID nits.

0.13. Changes from [draft-ietf-sieve-imapflags-03.txt](#)

1. Minor editorial changes from Ken.
2. Added IANA Considerations section.
3. Clarified "hasflag" behaviour with :matches, :contains, etc.
4. Added optional comparator field to "hasflag" test, for consistency with other extensions.
5. Clarified interaction of hasflag with the relational extension.

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 if a flag belongs to a list.

The capability string associated with extension defined in this document is "imap4flags".

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 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 [\[SIEVE\]](#).

2. Conventions used.

Conventions for notations are as in [\[SIEVE\] section 1.1](#), including use of [\[KEYWORDS\]](#) and "Usage:" label for the definition of action and tagged arguments syntax.

2.1. General requirements for flag handling

The following notes apply to processing of Addflag and Removeflag actions, hasflag test and :flags tagged argument.

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

The Sieve interpreter SHOULD check the list of flags for validity as described by [\[IMAP\]](#) ABNF. In particular non-ASCII characters are not

allowed in flag names. However spaces MUST be always allowed. Multiple spaces between flag names MUST be treated as a single space character, leading and trailing spaces MUST be ignored. A string containing a space-separated list of flags is equivalent to a string list consisting of the flags. The last requirement is to simplify amalgamation of multiple flag lists.

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 which 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 space-separated list of flag names.

Any of 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 MUST cause runtime error.

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 2 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, the last flag list value MUST win.

[3.1.](#) Setflag Action

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 "flags" "\\Deleted";
        }
```

A more substantial example is:

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

[3.2.](#) Addflag action

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 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](#). Removeflag Action

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 variable "MyFlags" has the value "A B", the following test

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

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

```
hasflag "MyFlags" ["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 test will evaluate to false:

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

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

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

alter 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 message to the target mailbox.

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

The copy of the message filed into 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.

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

[7.](#) Other Considerations

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

[8.](#) Security Considerations

Security considerations are discussed in the [[IMAP](#)], [[SIEVE](#)] and [[VARIABLES](#)].

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

9. 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

Capability keyword: imap4flags

Capability arguments: N/A

Standards Track/IESG-approved experimental RFC number:
this RFC

Person and email address to contact for further information:

Alexey Melnikov

<alexey.melnikov@isode.com>

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

10. 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 and 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";
    }

```

[11.](#) Acknowledgments

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

The help of those who took the time to review the draft 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 and Philip Guenther.

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

12. Author's Address

Alexey Melnikov
Isode Limited

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

Email: alexey.melnikov@isode.com

13. Normative References

[SIEVE] Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language", work in progress, I-D [draft-ietf-sieve-3028bis](#).

[KEYWORDS] Bradner, S., "Keywords for use in RFCs to Indicate Requirement Levels", Harvard University, [RFC 2119](#), March 1997.

[IMAP] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", University of Washington, [RFC 3501](#), March 2003.

[VARIABLES] Homme, K. T., "Sieve -- Variables Extension", University of Oslo, work in progress, [draft-ietf-sieve-variables-XX.txt](#)

[RELATIONAL] Leiba, B. and Segmuller, W., "Sieve Extension: Relational Tests", Work in Progress, [draft-ietf-sieve-3431bis-XX.txt](#)

14. Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights

in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

15. Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.