

Sieve Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 3, 2012

A. Melnikov
Isode Limited
Q. Sun
B. Leiba
K. Li
Huawei Technologies
December 1, 2011

Sieve Extension for Converting Messages Before Delivery
draft-ietf-sieve-convert-06

Abstract

This document describes how the "CONVERT" IMAP extension can be used within the Sieve mail filtering language to transform messages before final delivery.

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 June 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Conventions Used in this Document	3
2.	"convert" action	3
2.1.	Interaction with other tests and actions	4
2.2.	"convert" as a test	5
3.	Examples	5
3.1.	Example 1	5
3.2.	Example 2	6
3.3.	Example 3	6
3.4.	Example 4	7
4.	Security Considerations	8
5.	IANA Considerations	8
6.	Acknowledgements	8
7.	Normative References	8
	Authors' Addresses	9

1. Introduction

The IMAP "CONVERT" extension [RFC5259] adds an IMAP command for performing client-controlled conversions on whole messages or their body parts. This document defines a similar extension to the Sieve mail filtering language [RFC5228], which reuses the conversion parameters and framework established by IMAP CONVERT.

1.1. 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 RFC 2119 [RFC2119].

Conventions for notations are as in Sieve [RFC5228] section 1.1, including the use of ABNF [RFC5234].

2. "convert" action

```
Usage: convert <quoted-from-mime-type: string>
        <quoted-to-mime-type: string>
        <transcoding-params: string-list>
```

The "convert" action specifies that body parts with "quoted-from-mime-type" MIME type be converted to "quoted-to-mime-type" MIME type using conversion parameters specified in "transcoding-params". Each conversion parameter value has the following syntax: "<transcoding-param-name>=<transcoding-param-value>", where <transcoding-param-name> and <transcoding-param-value> are defined in CONVERT [RFC5259]. Messages that don't have any body parts with the "quoted-from-mime-type" MIME type are not affected by the conversion.

The "convert" action can be used with Sieve MIME Part Tests [RFC5703], in the case that some, but not all of the body parts need to be converted, or where different body parts might require different conversions. When the "convert" action appears in a "foreverypart" loop, it applies only to the body part being processed, and not to any other body parts (see Section 3.2 for an example).

When the "convert" action appears outside a "foreverypart" loop, the conversion applies equally to all body parts -- that is, all body parts that have the "quoted-from-mime-type" are converted, using the same transcoding parameters.

A single "convert" action will only apply once to any body part. If, for example, << convert "image/jpeg" "image/jpeg" ["pix-x=100","pix-

y=120"] >> converts a larger JPEG image to the smaller 100 x 120 size, that will be the end of that "convert" action on that body part. The action will not see a "new" JPEG body part to process, resulting from the conversion.

If a "convert" action cannot be completed -- perhaps because the conversion failed, or because the requested conversion is not available -- that "convert" action MUST terminate and leave the message unchanged, rolling back any other conversions done by that action. The script processing continues, and prior or subsequent "convert" actions are not affected. No error condition is raised, and no partial conversions from a single "convert" action are allowed.

Implementations might defer any actual conversion until the the results of the conversion are needed for script processing, to avoid doing conversions unnecessarily. Consider the case wherein a "convert" action is processed, but a "discard" action results without the need to actually perform the conversion.

When conversions actually need to be done, they can put a significant load on the server. Computationally expensive conversions of a lot of body parts can constitute an attack vector, and even if done legitimately they can create an unacceptable load. Servers MAY refuse conversions, or do them at lower priority, effectively slowing the requesting process in order to avoid negative effects on service to other processes.

2.1. Interaction with other tests and actions

Whether the actual conversion has been done yet or not, a successful "convert" action effectively changes the message, and all subsequent actions, including any other "convert" actions, apply to the changed message. The "convert" action does not affect the applicability of other actions; any action that was applicable before the "convert" is equally applicable to the changed message afterward.

When a disposition-type action, such as "fileinto" or "redirect", is encountered, the state of the message with respect to conversions is "locked in" for that disposition-type action. Whether the implementation performs the action at that point or batches it for later, it MUST perform the action on the message as it stood at the time, and MUST NOT include subsequent conversions encountered later in the script processing. Therefore, the sequence "convert, fileinto, convert, fileinto" will store two different versions of the message: the first "fileinto" uses only the first conversion, while the second uses both. See Section 3.4 for an example of how this can be used.

In addition, any tests done on the message and its parts will test the message after prior conversions have been done. The fourth block of Section 3.4 shows an example of this situation.

Convert actions are cumulative, and each conversion operates on the message as it stands after all prior conversions. See the fourth block of Section 3.4 for an example of how this might be tricky.

Because the implicit keep (see Section 2.10.2 of [RFC5228]), if it is in effect, acts on the final state of the message, all conversions are performed before any implicit keep.

2.2. "convert" as a test

To simplify testing for supported and successful conversions, the "convert" action can also be used as a test. As such, it will attempt to perform the requested conversion(s), and will evaluate to "false" if and only if at least one conversion failed. The failure can be because a conversion was unsupported, or because the data could not be converted (perhaps it had been corrupted in transit, or mislabeled at its origin).

This creates a new type of Sieve action, a "testable action". The usage as a test is exactly the same as for an action, and it doubles as an action and a test of the action's result at the same time. See Section 3.2 for an example of how this test can be used.

Note that defining this new testable action does not change the definitions of any other actions -- it does not imply that other actions can be used as tests. Future extensions might define other testable actions, but those specifications would be responsible for clearly specifying that.

3. Examples

3.1. Example 1

In the following example, all "image/tiff" body parts of the message are converted to "image/jpeg" with image resolution of 320x240 pixels. The converted message is then subject to the implicit keep.

```
require ["convert"];
convert "image/tiff" "image/jpeg" ["pix-x=320","pix-y=240"];
```

3.2. Example 2

In the following example, all "image/tiff" body parts of the message are converted to "image/jpeg", as in Example 1. If the conversions were successful, those messages are then filed into a mailbox called "INBOX.pics". Other messages (those with no image/tiff body parts) are subject to the implicit keep, and have not been converted.

```
require ["mime", "fileinto", "convert"];
if header :mime :anychild :contenttype
    "Content-Type" "image/tiff"
{
    if (convert "image/tiff" "image/jpeg" ["pix-x=320","pix-y=240"])
    {
        fileinto "INBOX.pics";
    }
}
```

3.3. Example 3

In the following example, only "image/tiff" body parts with a Content-Disposition of "inline" are converted. Matching parts that are larger than 500 kilobytes are converted using an image resolution of 640x480 pixels, and those smaller are converted to 320x240 pixels. The message disposition is not changed, so the implicit keep will be in effect unless something else in the script changes that.

```
require ["mime", "foreverypart", "fileinto", "convert"];
foreverypart
{
    if header :mime :param "filename" :contains
        "Content-Disposition" "inline"
    {
        if size :over "500K"
        {
            convert "image/tiff" "image/jpeg" ["pix-x=640","pix-y=480"];
        } else {
            convert "image/tiff" "image/jpeg" ["pix-x=320","pix-y=240"];
        }
    }
}

[... script continues ...]
```

3.4. Example 4

The following example shows some tricky interactions between multiple "convert" actions and other disposition-type actions.

```
require ["mime", "foreverypart",
        "fileinto", "redirect" "convert"];

# The first "if" block will convert all image/tiff body parts
# to 640x480 jpegs, and will file the message
# into the "INBOX.pics" mailbox as converted at this point.
if header :mime :anychild :contenttype
    "Content-Type" "image/tiff"
{
    convert "image/tiff" "image/jpeg" ["pix-x=640","pix-y=480"];
    fileinto "INBOX.pics";
}

# The second block, the "foreverypart" loop, will convert all
# inline jpegs to 320x240 resolution... including any tiff body
# parts that had been converted in the first block, above.
# Therefore, any tiff that had been converted to a 640x480 jpeg
# will be re-converted to a 320x240 jpeg here if its
# Content-Disposition is specified as "inline".
foreverypart
{
    if header :mime :param "filename" :contains
        "Content-Disposition" "inline"
    {
        convert "image/jpeg" "image/jpeg" ["pix-x=320","pix-y=240"];
    }
}

# The third block will take any message that contains a header
# field called "Mobile-Link" and redirect it to the user's
# mobile address. The redirected message will include both
# conversions above, from block one and block two.
if exists "Mobile-Link"
{
    redirect "joe@mobile.example.com";
}

# The fourth block will file the message into "Tiff" if it
# contains any tiff body parts. But because of the earlier
# conversion (in the first block), there will never be any
# tiff body parts, so this "fileinto" will never happen.
if header :mime :anychild :contenttype
    "Content-Type" "image/tiff"
```

```
{  
  fileinto "Tiff";  
}
```

```
# Now, at the end of the script processing, the Sieve  
# processor will perform an implicit keep if none of  
# the "fileinto" and "redirect" actions were taken.  
# The kept message will include any conversions that  
# were done (that is, any from the second block).
```

4. Security Considerations

Security considerations given in IMAP CONVERT [RFC5259] and Sieve [RFC5228] are relevant to this document. There are no additional security considerations resulting from combining the two.

5. IANA Considerations

IANA is requested to add the following registration to the Sieve Extensions registry, as defined in RFC 5228:

Capability name: convert

Description: adds a new Sieve test and action that enable Sieve scripts to perform data conversions on the message being delivered.

RFC number: this RFC

Contact address: The Sieve discussion list <sieve@ietf.org>

6. Acknowledgements

The authors also want to thank all who have contributed key insight and extensively reviewed and discussed the concepts of CONVERT.

7. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5228] Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language", RFC 5228, January 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

- [RFC5259] Melnikov, A. and P. Coates, "Internet Message Access Protocol - CONVERT Extension", RFC 5259, July 2008.
- [RFC5703] Hansen, T. and C. Daboo, "Sieve Email Filtering: MIME Part Tests, Iteration, Extraction, Replacement, and Enclosure", RFC 5703, October 2009.

Authors' Addresses

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com
URI: <http://www.melnikov.ca/>

Qian Sun
Huawei Technologies
Bantian Longgang
Shenzhen, Guangdong 518129
P.R China

Phone: +86 755 28780808
Email: sunqian@huawei.com

Barry Leiba
Huawei Technologies

Phone: +1 646 827 0648
Email: barryleiba@computer.org
URI: <http://internetmessagingtechnology.org/>

Kepeng Li
Huawei Technologies
Huawei Base, Bantian, Longgang District
Shenzhen, Guangdong 518129
P. R. China

Phone: +86-755-28974289
Email: likepeng@huawei.com

Sieve Working Group
Internet-Draft
Updates: 5228 (if approved)
Intended status: Standards Track
Expires: March 18, 2013

B. Leiba
Huawei Technologies
September 14, 2012

Support for Internet Message Access Protocol (IMAP) Events in Sieve
draft-ietf-sieve-imap-sieve-09

Abstract

Sieve defines an email filtering language that can, in principle, plug into any point in the processing of an email message. As defined in the base specification, it plugs into mail delivery. This document defines how Sieve can plug into points in the IMAP protocol where messages are created or changed, adding the option of user-defined or installation-defined filtering (or, with Sieve extensions, features such as notifications). Because this requires future Sieve extensions to specify their interactions with this one, this document updates the base Sieve specification, RFC 5228.

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 March 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	4
1.1.	Overview	4
1.2.	Differences Between IMAP Events and Mail Delivery	4
1.3.	Conventions used in this document	5
2.	The IMAP Events in Sieve Extension	6
2.1.	The "imapsieve" Capability Strings	6
2.2.	Existing IMAP Functions Affected by IMAP events in Sieve	6
2.2.1.	The IMAP APPEND Command	7
2.2.2.	The IMAP COPY Command	7
2.2.3.	Changes to IMAP Message Flags	7
2.2.4.	When Script Actions Set the \Deleted Flag	8
2.3.	New Functions Defined by IMAP events in Sieve	8
2.3.1.	Interaction with Metadata	8
3.	Applicable Sieve Actions and Interactions	10
3.1.	The Implicit Keep	10
3.2.	The Keep Action	10
3.3.	The Fileinto Action	10
3.4.	The Redirect Action	11
3.5.	The Discard Action	11
3.6.	The Notify Action	11
3.7.	The Addheader and Deleteheader Actions	12
3.8.	The Setflag, Deleteflag, and Removeflag Actions	12
3.9.	MIME Part Tests and Replacement	12
3.10.	Spamtest and Virustest	12
3.11.	Inapplicable Actions	12
3.12.	Future Sieve Actions	13
4.	Interaction With Sieve Environment	14
4.1.	Base Sieve Environment Items: location and phase	14
4.2.	New Sieve Environment Items: imap.user and imap.email	14
4.3.	New Sieve Environment Item: imap.cause	14
4.4.	New Sieve Environment Item: imap.mailbox	15
4.5.	New Sieve Environment Item: imap.changedflags	15
4.6.	Interaction With Sieve Tests (Comparisons)	15
5.	Examples	16

6.	Security Considerations	17
7.	IANA Considerations	18
7.1.	Registration of "imapsieve" IMAP capability	18
7.2.	Registration of "imapsieve" Sieve extension	18
7.3.	Registration of Sieve Environment Items	18
7.3.1.	Registration of Sieve Environment Item: imap.cause	18
7.3.2.	Registration of Sieve Environment Item: imap.mailbox . . .	19
7.3.3.	Registration of Sieve Environment Item: imap.changedflags	19
7.3.4.	Registration of Sieve Environment Item: imap.user	19
7.3.5.	Registration of Sieve Environment Item: imap.email	19
7.4.	Registration of IMAP METADATA Mailbox Entry Name	19
7.5.	Registration of IMAP METADATA Server Entry Name	20
8.	References	21
8.1.	Normative References	21
8.2.	Informative References	21
	Author's Address	23

1. Introduction

1.1. Overview

Some applications have a need to apply Sieve filters [RFC5228] in contexts other than initial mail delivery. This is especially true in diverse service environments, such as when the client is sporadically connected, is connected through a high-latency or high-cost channel, or is on a limited-function device. For such clients, it may be very important, for higher performance and reliability, to take advantage of server capabilities, including those provided by Sieve filtering (and Sieve extensions, such as Notify [RFC5435]).

This specification defines extensions to IMAP [RFC3501] to support the invocation of Sieve scripts at times when the IMAP server creates new messages or modifies existing ones. It also defines how Sieve scripts will process these invocations. Support for IMAP events in Sieve also requires support for the following:

- o IMAP Metadata [RFC5464], because Metadata is used to associate scripts with IMAP mailboxes.
- o Sieve Environment [RFC5183], because it defines an important way for Sieve scripts to test the conditions under which they have been invoked.
- o Sieve IMAP4Flags [RFC5232], because it provides important functionality in handling IMAP events related to flag changes.

Because this requires future Sieve extensions to specify their interactions with this one (see Section 3.12), this document updates the base Sieve specification, RFC 5228.

1.2. Differences Between IMAP Events and Mail Delivery

Invoking Sieve scripts in a context other than initial mail delivery introduces new situations, which changes the applicability of Sieve features and creates implementation challenges and user interface issues. This section discusses some of those differences, challenges, and issues.

At times other than message delivery, delivery "envelope" information might not be available. With messages added through IMAP APPEND, there might be no way to even guess who the intended recipient is, and no concept of who "sent" the message. Sieve actions that relate to contacting the sender, for example, will not be applicable.

Because IMAP events will often be triggered by user actions, and

because user interfaces allow bulk actions that differ from individual message arrival, it now becomes possible for a single user action, such as drag-and-drop, to initiate Sieve script processing on a large number of messages at once. Implementations will have to deal with such situations as a "COPY" action or flag changes on dozens, or even thousands of messages.

Other issues might surface as this extension is deployed and experience with it develops.

1.3. 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 [RFC2119].

2. The IMAP Events in Sieve Extension

2.1. The "imapsieve" Capability Strings

An IMAP server advertises support for IMAP events in Sieve through the "imapsieve" capability. A server that advertises "imapsieve" is claiming to be in compliance with this specification in all aspects. The syntax of the "imapsieve" capability string is defined as follows:

```
capability /= "IMAPSIEVE=" sieveurl-server
               ; <sieveurl-server> is defined in RFC 5804, Section 3
```

Only one "imapsieve" capability string, specifying one sieveurl-server, is allowed to be present. The sieveurl-server identifies the ManageSieve server that clients need to contact for managing Sieve scripts associated with this IMAP server.

The corresponding Sieve implementation uses the Sieve capability string "imapsieve", and Sieve scripts that depend upon the IMAP events MUST include that string in their "required" lists.

Implementations that support IMAP events in Sieve MUST also support IMAP Metadata [RFC5464] and Sieve Environment [RFC5183], because Metadata is used to associate scripts with IMAP mailboxes and Environment defines an important way for Sieve scripts to test the conditions under which they have been invoked. Notwithstanding the support requirement, scripts that directly use Environment MUST also include its capability string in their "required" lists.

2.2. Existing IMAP Functions Affected by IMAP events in Sieve

The subsections below describe in detail the IMAP commands and situations on which IMAP events in Sieve have an effect. Not all Sieve actions make sense in the case of messages affected by IMAP commands. See Section 3 for details.

It's important to note that since the base Sieve specification (see [RFC5228]) and its extensions define functions for scripts that are invoked during initial mail delivery, those function definitions are necessarily tailored to and limited by that context. This document extends those function definitions for use during IMAP events. By nature of that, Sieve functions, in this extended context, may behave somewhat differently, though their extended behaviour will still be consistent with the functions' goals.

If more than one message is affected at the same time, each message triggers the execution of a Sieve script separately. The scripts MAY

be run in parallel.

2.2.1. The IMAP APPEND Command

A message may be added to a mailbox through the IMAP APPEND command. In a server that advertises "imapsieve", new messages added in this way MUST trigger the execution of a Sieve script, subject to the settings defined through Metadata (see Section 2.3.1).

If the IMAP server also supports the IMAP MultiAppend extension [RFC3502], the APPEND command can create more than one message at a time. In that case, each message creation is considered a separate event, and any applicable Sieve script is called once for each message.

2.2.2. The IMAP COPY Command

One or more messages may be added to a mailbox through the IMAP COPY command. In a server that advertises "imapsieve", new messages added in this way MUST trigger the execution of a Sieve script, subject to the settings defined through Metadata.

2.2.3. Changes to IMAP Message Flags

One or more existing messages can have their flags changed in a number of ways, including:

- o The FETCH command (may cause the \Seen flag to be set).
- o The STORE command (may cause the \Answered, \Deleted, \Draft, \Flagged, and \Seen flags to be set or reset, and may cause keywords to be set or reset).
- o The invocation of a Sieve script on an existing message, where the script uses one of the actions defined in the IMAP4Flags extension [RFC5232] to change the flags.

In a server that advertises "imapsieve", messages whose flags are changed in any way (except as explained in the next sentence) MUST trigger the execution of a Sieve script, subject to the settings defined through Metadata. The exception is that in order to avoid script loops, flag changes that are made as a result of a script that was itself invoked because of flag changes SHOULD NOT result in a further invocation of the script. In any case, implementations MUST take steps to avoid such loops.

For flag-change events, the Sieve script will see the message flags as they are AFTER the changes.

2.2.4. When Script Actions Set the \Deleted Flag

There are times when the actions "fileinto" (see Section 3.3), "redirect" (see Section 3.4), and "discard" (see Section 3.5) will set the \Deleted flag on the message. In those cases, the following apply:

When the \Deleted flag is set by the script, a flag-change Sieve script is not invoked.

The implementation MAY then expunge the original message (WITHOUT expunging other messages in the mailbox). Alternatively, it might have expunges batched or done by a user. (It might be helpful to allow the user to make this choice through a preference.)

If the server does the expunge, the effect is as though a client had flagged the message and done a UID EXPUNGE (see [RFC4315]) on the affected message(s) only. Handling it this way allows clients to handle messages consistently, and avoids hidden changes that might invalidate their message caches.

2.3. New Functions Defined by IMAP events in Sieve

2.3.1. Interaction with Metadata

Support for IMAP events in Sieve requires support for IMAP Metadata [RFC5464] as well, since the latter is used to associate scripts with IMAP mailboxes.

When an applicable event occurs on an IMAP mailbox, if there is an IMAP metadata entry named "/shared/imap sieve/script" for the mailbox, that entry is used. If there is not, but there is an IMAP metadata entry named "/shared/imap sieve/script" for the server, that entry is used (providing a way to define a global script for all mailboxes on a server). If neither entry exists, then no script will be invoked.

If a "/shared/imap sieve/script" metadata entry was selected above, its value is used as the name of the Sieve script that will be invoked in response to the IMAP event. If the value is empty, then no script is run. The selection of which metadata entry to use happens before any examination of the contents of the entry. If the mailbox entry is selected and is then found to be unusable or empty, the server entry is not used as a backup: no script is run.

This specifies the mechanism for "activating" a script for a given mailbox (or for all mailboxes), but does not specify a mechanism for creating, storing, or validating the script. Implementations MUST support ManageSieve [RFC5804], and can use the PUTSCRIPT command to

store the script without using the SETACTIVE command to activate it.

Script names used in `"/shared/imap sieve/script"` metadata entries are the script names used on the corresponding ManageSieve server. If a `"/shared/imap sieve/script"` metadata entry contains a script name that doesn't exist in the ManageSieve server, then no Sieve script will be invoked for IMAP Sieve events.

Only one Sieve script may currently be defined per mailbox, eliminating the complexity and possible ambiguity involved with coordinating the results of multiple scripts. Any sub-filtering is done in the Sieve script. For example, if it's only necessary to deal with flag changes, but not with new messages appended or copied, the Sieve script will still be invoked for all events, and the script is responsible for checking the event type.

The possibility is open for an extension to add support for multiple scripts -- for example, per-client scripts on a multi-client user's inbox, or per-user scripts on a mailbox that is shared among users.

Because this metadata name is associated with the mailbox, there can (and it's expected that there will) be different scripts associated with events for different mailboxes. Indeed, most mailboxes will probably invoke no script at all.

3. Applicable Sieve Actions and Interactions

Since some Sieve actions relate specifically to the delivery of mail, not all actions and extensions make sense when the messages are created by other means or when changes are made to data associated with existing messages. This section describes how actions in the base Sieve specification, and those in extensions known at this writing, relate to this specification.

In addition to what is specified here, interactions noted in the individual specifications apply, and must be considered.

3.1. The Implicit Keep

For all cases that fall under IMAP events in Sieve, the implicit keep means that the message is treated as it would have been if no Sieve script were run. For APPEND and COPY, the message is stored into the target mailbox normally. For flag changes, the message is left in the mailbox. If actions have been taken that change the message, those changes are considered transient and MUST NOT be retained for any keep action (because IMAP messages are immutable). No error is generated, but the original message, without the changes, is kept.

3.2. The Keep Action

The keep action is applicable in all cases that fall under IMAP events in Sieve. Its behaviour is as described for implicit keep, in Section 3.1.

3.3. The Fileinto Action

If the Sieve implementation supports the fileinto action, that action is applicable in all cases that fall under IMAP events in Sieve. If the Copy extension [RFC3894] is available and the :copy option is specified, the implicit keep is retained; otherwise, fileinto cancels the implicit keep, as specified in the base Sieve specification.

For APPEND and COPY, the message is stored into the fileinto mailbox IN ADDITION TO the original target mailbox. For flag changes, the message is COPIED into the fileinto mailbox, without removing the original. In all cases, fileinto always creates a new message, separate from the original.

The fileinto action is not an IMAP APPEND or COPY, and therefore does not result in a subsequent event (see also the Security Considerations, Section 6).

If a keep action is not also in effect, the original message is then

marked with the \Deleted flag (see Section 2.2.4).

3.4. The Redirect Action

The redirect action is applicable in all cases that fall under IMAP events in Sieve. It causes the message to be sent, as specified in the base Sieve specification, to the designated address. If the Copy extension [RFC3894] is available and the :copy option is specified, the implicit keep is retained; otherwise, redirect cancels the implicit keep, as specified in the base Sieve specification.

It's possible that a message processed in this way does not have the information necessary to be redirected properly. It might lack necessary header information, and there might not be appropriate information for the MAIL FROM command. In such cases, the "redirect" action uses Message Submission [RFC6409], and it is up to the Sieve engine to supply the missing information. The redirect address is, of course, used for the "RCPT TO", and the "MAIL FROM" SHOULD be set to the address of the owner of the mailbox. The message submission server is allowed, according to the Message Submission protocol, to perform necessary fix-up to the message (see Section 8 of RFC 6409). It can also reject the submission attempt, if the message is too ill-formed for submission.

For APPEND and COPY, the message is stored into the target mailbox in addition to being redirected. For flag changes, the message remains in its original mailbox.

If a keep action is not also in effect, the original message is then marked with the \Deleted flag (see Section 2.2.4).

3.5. The Discard Action

The discard action is applicable in all cases that fall under IMAP events in Sieve. For APPEND and COPY, the message is first stored into the target mailbox. If an explicit keep action is also in effect, the discard action now does nothing. Otherwise, the original message is then marked with the \Deleted flag (see Section 2.2.4).

3.6. The Notify Action

If the Notify extension [RFC5435] is available, the notify action is applicable in all cases that fall under IMAP events in Sieve. The result is that the requested notification is sent, and that the message is otherwise handled as it would normally have been.

3.7. The Addheader and Deleteheader Actions

If the EditHeader extension [RFC5293] is available, it can be used to make transient changes to header fields, which aren't saved in place, such as for "redirect" or "fileinto" actions. Because messages in IMAP mailboxes are immutable, such changes are not applicable for the "keep" action (explicit or implicit). See Section 3.1.

3.8. The Setflag, Deleteflag, and Removeflag Actions

Implementations of IMAP events in Sieve MUST also support the IMAP4Flags extension [RFC5232], and the actions associated with it are all applicable to any case that falls under IMAP events in Sieve.

It is worth noting also that the "hasflag" test that is defined in the IMAP4Flags extension might be particularly useful in scripts triggered by flag changes ("hasflag" will see the new, changed flags). The flag changes behave as though a client had made the change.

As explained above, in order to avoid script loops flag changes that are made as a result of a script that was itself invoked because of flag changes SHOULD NOT result in another script invocation. In any case, implementations MUST take steps to avoid such loops.

3.9. MIME Part Tests and Replacement

If the MIME Part Tests extension [RFC5703] is available, all of its functions can be used, but any changes made to the message, using the "replace" or "enclose" action, MUST be considered transient, and are only applicable with actions such as "redirect" and "fileinto". Because messages in IMAP mailboxes are immutable, such changes are not applicable for the "keep" action (explicit or implicit). See Section 3.1.

3.10. Spamtest and Virustest

If the Spamtest and Virustest extensions [RFC5235] are available, they are applicable in all cases that fall under IMAP events in Sieve.

3.11. Inapplicable Actions

The following actions and extensions are not applicable to any case that falls under IMAP events in Sieve, because they are specifically designed to respond to delivery of a new email message. Their appearance in the "require" control or their use in an IMAP event MUST result in an error condition that will terminate the Sieve

script:

reject [RFC5228]

ereject [RFC5429]

vacation [RFC5230]

Future extensions that are specifically designed to respond to delivery of a new email message will likewise not be applicable to this extension.

3.12. Future Sieve Actions

As noted above, future extensions that are specifically designed to respond to delivery of a new email message will not be applicable to this extension, because this extension does not involve acting at new-message delivery time.

In general, future extensions to Sieve that define new actions **MUST** specify the applicability of those actions to this specification.

4. Interaction With Sieve Environment

4.1. Base Sieve Environment Items: location and phase

The Sieve Environment extension defines a set of standard environment items (see [RFC5183], Section 4.1). Two of those items are affected when the script is invoked through an IMAP event.

The value of "location" is set to "MS" -- evaluation is being performed by a Message Store.

The value of "phase" is set to "post" -- processing is taking place after (or perhaps instead of, in the case of APPEND) final delivery.

4.2. New Sieve Environment Items: imap.user and imap.email

In the normal case, when Sieve is used in final delivery, there is no identity for the "filer" -- the user who is creating or changing the message. In this case, there is such an identity, and a Sieve script might want to access that identity.

Implementations MUST set and make available two new environment items:

"imap.user" -- the identity (login ID) of the IMAP user that caused the action. This MUST be the empty string if it is accessed during normal (final delivery) Sieve processing.

"imap.email" -- the primary email address of the IMAP user that caused the action (the user identified by "imap.user"). In some implementations, "imap.user" and "imap.email" might have the same value. This MUST be the empty string if it is accessed during normal (final delivery) Sieve processing.

4.3. New Sieve Environment Item: imap.cause

Each mailbox uses a single script for all the change conditions described in this document (append, copy, flag changes). To support that, the implementation MUST set the Environment [RFC5183] item "imap.cause", which contains the name of the action that caused the script to be invoked. Its value is one of the following:

- o APPEND (for invocations resulting from APPEND commands)
- o COPY (for invocations resulting from COPY commands)
- o FLAG (for invocations resulting from flag changes)

Future extensions might define new events and, thus, new causes. Such extensions will come with their own capability strings, and the events they define will only be presented when their capabilities are requested. Scripts that do not request those capabilities will not see those events, and will not encounter the new cause strings.

4.4. New Sieve Environment Item: `imap.mailbox`

The implementation MUST set the Environment [RFC5183] item `"imap.mailbox"` to the name of the mailbox that the affected message is in, in the case of existing messages, or is targeted to be stored into, in the case of new messages. The value of this item is fixed when the script begins, and, in particular, MUST NOT change as a result of any action, such as `"fileinto"`.

4.5. New Sieve Environment Item: `imap.changedflags`

If the script was invoked because of flag changes to an existing message, the implementation MUST set the Environment [RFC5183] item `"imap.changedflags"` to the name(s) of the flag(s) that have changed. If the script was not invoked because of flag changes, the value of this item MUST be the empty string. The script will not know from this item whether the flags have been set or reset, but it can use the `"hasflag"` test to determine the current value. See example 2 in Section 5 for an example of how this might be used.

4.6. Interaction With Sieve Tests (Comparisons)

Any tests against message envelope information, including the `"envelope"` test in the Sieve base specification, as well as any such test defined in extensions, are either inapplicable or have serious interoperability issues when performed at other than final-delivery time. Therefore, envelope tests MUST NOT be permitted in the cases described here, and their use MUST generate a runtime error.

This extension does not affect the operation of other tests or comparisons in the Sieve base specification.

5. Examples

Example 1:

If a new message is added to the "ActionItems" mailbox, a copy is sent to the address "actionitems@example.com".

```
require ["copy", "environment", "imapsieve"];

if anyof (environment :is "imap.cause" "APPEND",
          environment :is "imap.cause" "COPY") {
    if environment :is "imap.mailbox" "ActionItems" {
        redirect :copy "actionitems@example.com";
    }
}
```

Example 2:

If the script is called for any message with the \Flagged flag set (tested through the IMAP4Flags extension [RFC5232]) AND this script invocation represents a change to that flag, then a notification is sent using the Notify extension [RFC5435]. No notification will be sent, though, if we're called with an existing message that already had that flag set.

```
require ["enotify", "imap4flags", "variables",
          "environment", "imapsieve"];

if environment :matches "imap.mailbox" "*" {
    set "mailbox" "${1}";
}

if allof (hasflag "\\Flagged",
          environment :contains "imap.changedflags" "\\Flagged") {
    notify :message "Important message in ${mailbox}"
           "xmpp:tim@example.com?message;subject=SIEVE";
}
```

Example 3:

This shows an example IMAP CAPABILITY response when this extension is supported. The client has done STARTTLS with the server, and is now inspecting capabilities. (The untagged CAPABILITY response is split here for readability only, but will be in one response message.)

```
C: A01 CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=PLAIN UIDPLUS LIST-EXTENDED
   ACL IMAPSIEVE=sieve://sieve.example.com MULTISEARCH
S: A01 OK done
```

6. Security Considerations

It is possible to introduce script processing loops by having a Sieve script that is triggered by flag changes use the actions defined in the IMAP4Flags extension [RFC5232]. Implementations **MUST** take steps to prevent script loops. One way to avoid this problem is that if a script is invoked by flag changes, and that script further changes the flags, those flag changes **SHOULD NOT** trigger a Sieve script invocation.

The `fileinto` action never results in the invocation of a script. If an implementation did invoke a script as the result of a `fileinto`, as though an IMAP APPEND or COPY had been done, script loops could result (mailbox A responds to all COPY events by doing "fileinto B", and mailbox B responds to all COPY events by doing "fileinto A"). In general, actions taken as a result of the Sieve script are not IMAP events, and do not themselves trigger Sieve script invocations.

It is also possible to introduce loops through the "redirect" or "notify" actions. See Section 10 of Sieve [RFC5228], Section 8 of Sieve Notify [RFC5435], and the Security Considerations sections of the applicable notification-method documents for loop-prevention information. This extension does not change any of that advice.

This extension introduces side-effects to IMAP commands that users and script authors might not be aware of and that can accidentally be triggered by an operation that the user would expect to be innocuous. In particular, certain actions, such as `redirect`, can cause a message (such as a message appended to a mailbox by a user) to be sent to the Internet in response to something as simple as a flag change. For example, a script might cause messages marked for deletion to be sent to some off-site archiving service. If a user appends a draft message to a mailbox (perhaps an unencrypted draft message) and then marks it for deletion, it might be very surprising to the user that the message is sent off site. Script authors need to be careful not to create these kinds of surprises, especially when creating global scripts.

Other security considerations are discussed in IMAP [RFC3501], and Sieve [RFC5228], as well as in some of the other extension documents.

7. IANA Considerations

7.1. Registration of "imapsieve" IMAP capability

IANA is asked to add "IMAPSIEVE=" to the IMAP 4 Capabilities registry, according to the IMAP 4 specification [RFC3501]. (<http://www.iana.org/assignments/imap4-capabilities>)

7.2. Registration of "imapsieve" Sieve extension

The following information should be added to the Sieve Extensions registry, according to the Sieve specification [RFC5228]. (<http://www.iana.org/assignments/sieve-extensions/sieve-extensions.xml>)

To: iana@iana.org
Subject: Registration of new Sieve extension
Capability name: imapsieve
Description: Add Sieve processing for IMAP events.
RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

7.3. Registration of Sieve Environment Items

The following subsections register items in the Sieve Environment Items registry, according to the Environment extension [RFC5183]. (<http://www.iana.org/assignments/sieve-environment-items/sieve-environment-items.xml>)

7.3.1. Registration of Sieve Environment Item: imap.cause

To: iana@iana.org
Subject: Registration of new Sieve environment item
Item name: imap.cause
Description: The name of the action that caused the script to be invoked. Its value is one of the following:

- o APPEND (for invocations resulting from APPEND commands)
- o COPY (for invocations resulting from COPY commands)
- o FLAG (for invocations resulting from flag changes)

RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

7.3.2. Registration of Sieve Environment Item: imap.mailbox

To: iana@iana.org
Subject: Registration of new Sieve environment item
Item name: imap.mailbox
Description: The name of the mailbox that the affected message is in, in the case of existing messages, or is targeted to be stored into, in the case of new messages. The value of this item is fixed when the script begins, and, in particular, MUST NOT change as a result of any action, such as "fileinto".
RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

7.3.3. Registration of Sieve Environment Item: imap.changedflags

To: iana@iana.org
Subject: Registration of new Sieve environment item
Item name: imap.changedflags
Description: If the script was invoked because of flag changes to an existing message, this contains the name(s) of the flag(s) that have changed. Otherwise, the value of this item MUST be the empty string.
RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

7.3.4. Registration of Sieve Environment Item: imap.user

To: iana@iana.org
Subject: Registration of new Sieve environment item
Item name: imap.user
Description: The identity (IMAP login ID) of the IMAP user that caused the action.
RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

7.3.5. Registration of Sieve Environment Item: imap.email

To: iana@iana.org
Subject: Registration of new Sieve environment item
Item name: imap.email
Description: The primary email address of the IMAP user that caused the action (the user identified by "imap.user").
RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

7.4. Registration of IMAP METADATA Mailbox Entry Name

The following information should be added to the IMAP METADATA Mailbox Entry Registry, according to the Metadata extension

[RFC5464].

(<http://www.iana.org/assignments/imap-metadata/imap-metadata.xml>)

To: iana@iana.org
Subject: IMAP METADATA Entry Registration
Type: Mailbox
Name: /shared/impsieve/script
Description: This entry name is used to define mailbox metadata associated with IMAP events in Sieve for the associated mailbox. Specifically, this specifies the Sieve script that will be invoked when IMAP events occur on the specified mailbox.
Content-type: text/plain; charset=utf-8
RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

7.5. Registration of IMAP METADATA Server Entry Name

The following information should be added to the IMAP METADATA Server Entry Registry, items according to the Metadata extension [RFC5464]. (<http://www.iana.org/assignments/imap-metadata/imap-metadata.xml>)

To: iana@iana.org
Subject: IMAP METADATA Entry Registration
Type: Server
Name: /shared/impsieve/script
Description: This entry name is used to define metadata associated globally with IMAP events in Sieve for the associated server. Specifically, this specifies the Sieve script that will be invoked when IMAP events occur on any mailbox in the server that does not have its own mailbox-level /shared/impsieve/script entry.
Content-type: text/plain; charset=utf-8
RFC number: [[this RFC]]
Contact address: Sieve mailing list <sieve@ietf.org>

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.
- [RFC3502] Crispin, M., "Internet Message Access Protocol (IMAP) - MULTIAPPEND Extension", RFC 3502, March 2003.
- [RFC3894] Degener, J., "Sieve Extension: Copying Without Side Effects", RFC 3894, October 2004.
- [RFC5183] Freed, N., "Sieve Email Filtering: Environment Extension", RFC 5183, May 2008.
- [RFC5228] Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language", RFC 5228, January 2008.
- [RFC5232] Melnikov, A., "Sieve Email Filtering: Imap4flags Extension", RFC 5232, January 2008.
- [RFC5464] Daboo, C., "The IMAP METADATA Extension", RFC 5464, February 2009.
- [RFC5804] Melnikov, A. and T. Martin, "A Protocol for Remotely Managing Sieve Scripts", RFC 5804, July 2010.
- [RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, November 2011.

8.2. Informative References

- [RFC4315] Crispin, M., "Internet Message Access Protocol (IMAP) - UIDPLUS extension", RFC 4315, December 2005.
- [RFC5230] Showalter, T. and N. Freed, "Sieve Email Filtering: Vacation Extension", RFC 5230, January 2008.
- [RFC5235] Daboo, C., "Sieve Email Filtering: Spamtest and Virustest Extensions", RFC 5235, January 2008.
- [RFC5293] Degener, J. and P. Guenther, "Sieve Email Filtering: Editheader Extension", RFC 5293, August 2008.

- [RFC5429] Stone, A., "Sieve Email Filtering: Reject and Extended Reject Extensions", RFC 5429, March 2009.
- [RFC5435] Melnikov, A., Leiba, B., Segmuller, W., and T. Martin, "Sieve Email Filtering: Extension for Notifications", RFC 5435, January 2009.
- [RFC5703] Hansen, T. and C. Daboo, "Sieve Email Filtering: MIME Part Tests, Iteration, Extraction, Replacement, and Enclosure", RFC 5703, October 2009.

Author's Address

Barry Leiba
Huawei Technologies

Phone: +1 646 827 0648

Email: barryleiba@computer.org

URI: <http://internetmessagingtechnology.org/>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 3, 2012

C. Daboo
A. Stone
January 31, 2012

Sieve Email Filtering: Include Extension
draft-ietf-sieve-include-15

Abstract

The Sieve Email Filtering "include" extension permits users to include one Sieve script inside another. This can make managing large scripts or multiple sets of scripts much easier, and allows a site and its users to build up libraries of scripts. Users are able to include their own personal scripts or site-wide scripts.

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 August 3, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction and Overview	3
2. Conventions Used in This Document	3
3. Include Extension	3
3.1. General Considerations	3
3.2. Control Structure include	5
3.3. Control Structure return	8
3.4. Interaction with Variables Extension	8
3.4.1. Control Structure global	8
3.4.2. Variables Namespace global	10
3.5. Interaction with Other Extensions	11
4. Security Considerations	12
5. IANA Considerations	12
5.1. "include" Extension Registration	13
6. References	13
6.1. Normative References	13
6.2. Informative References	13
Appendix A. Acknowledgments	14
Appendix B. Change History (to be removed prior to publication as an RFC)	14
Authors' Addresses	17

1. Introduction and Overview

It's convenient to be able to break SIEVE [RFC5228] scripts down into smaller components which can be reused in a variety of different circumstances. For example, users may want to have a default script and a special 'vacation' script, the latter being activated when the user goes on vacation. In that case the default actions should continue to be run, but a vacation command should be executed first. One option is to edit the default script to add or remove the vacation command as needed. Another is to have a vacation script that simply has a vacation command and then includes the default script.

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 [RFC2119].

Conventions for notations are as in SIEVE [RFC5228] Section 1.1.

The following key phrases are used to describe scripts and script execution:

script

a valid Sieve script.

script execution

an instance of a Sieve interpreter invoked for a given message delivery, starting with the user's active script and continuing through any included scripts until the final disposition of the message (e.g. delivered, forwarded, discarded, rejected, etc.).

immediate script

the individual Sieve script file being executed.

including script

the individual Sieve script file that had an include statement which included the immediate script.

3. Include Extension

3.1. General Considerations

Sieve implementations that implement the "include", "return", and "global" commands described below have an identifier of "include" for

use with the capability mechanism. If any of the "include", "return", or "global" commands are used in a script, the "include" capability MUST be listed in the "require" statement in that script.

Sieve implementations need to track the use of actions in included scripts so that implicit "keep" behavior can be properly determined based on whether any actions have executed in any script.

Sieve implementations are allowed to limit the total number of nested included scripts, but MUST provide for a total of at least three levels of nested scripts including the top-level script. An error MUST be generated either when the script is uploaded to the Sieve repository, or when the script is executed, if any nesting limit is exceeded. If such an error is detected whilst processing a Sieve script, an implicit "keep" action MUST be executed to prevent loss of any messages.

Sieve implementations MUST NOT allow recursive script inclusion. Both direct recursion, where script A includes script A (itself), and indirect recursion, where script A includes script B which includes script A once again, are prohibited.

Sieve implementations MUST generate an error at execution time if an included script is a recursive inclusion. Implementations MUST NOT generate errors for recursive includes at upload time, as this would force an upload ordering requirement upon script authors and generators.

Sieve implementations MUST generate an error at execution time if an included script does not exist, except when the ":optional" parameter is specified. Implementations MUST NOT generate errors for scripts missing at upload time, as this would force an upload ordering requirement upon script authors and generators.

If the Sieve "variables" extension [RFC5229] is present, an issue arises with the "scope" of variables defined in scripts that may include each other. For example, if a script defines the variable "\${status}" with one particular meaning or usage, and another defines "\${status}" with a different meaning, then if one script includes the other there is an issue as to which "\${status}" is being referenced. To solve this problem, Sieve implementations MUST follow the scoping rules defined in Section 3.4 and support the "global" command defined there.

3.2. Control Structure include

Usage: include [LOCATION] [":once"] [":optional"] <value: string>

LOCATION = ":personal" / ":global"

The "include" command takes an optional "location" parameter, an optional ":once" parameter, an optional ":optional" parameter, and a single string argument representing the name of the script to include for processing at that point. Implementations MUST restrict script names according to MANAGESIEVE [RFC5804], Section 1.6. The script name argument MUST be a constant string as defined in VARIABLES [RFC5229], Section 3; implementations MUST NOT expand variables in the script name argument.

The "location" parameter MUST default to ":personal" if not specified. The "location" parameter MUST NOT be specified more than once. The "location" has the following meanings:

:personal

Indicates that the named script is stored in the user's own personal (private) Sieve repository.

:global

Indicates that the named script is stored in a site-wide Sieve repository, accessible to all users of the Sieve system.

The ":once" parameter tells the interpreter only to include the named script if it has not already been included at any other point during script execution. If the script has already been included, processing continues immediately following the include command. Implementations MUST NOT generate an error if an "include :once" command names a script whose inclusion would be recursive; in this case, the script MUST be considered previously included and therefore "include :once" will not include it again.

Note: It is RECOMMENDED that script authors and generators use the ":once" parameter only when including a script that performs general duties such as declaring global variables and making sanity checks of the environment.

The ":optional" parameter indicates that the script may be missing. Ordinarily, an implementation MUST generate an error during execution if an include command specifies a script that does not exist. When ":optional" is specified, implementations MUST NOT generate an error for a missing script, and MUST continue as if the include command had not been present.

The included script **MUST** be a valid Sieve script. Implementations **MUST** validate that each script has its own "require" statements for all optional capabilities used by that script. The scope of a "require" statement is the script in which it immediately appears, and neither inherits nor passes on capabilities to other scripts during the course of execution.

A "stop" command in an included script **MUST** stop all script processing, including the processing of the scripts that include the immediate one. The "return" command (described below) stops processing of the immediate script only, and allows the scripts that include it to continue.

The "include" command **MAY** appear anywhere in a script where a control structure is legal, and **MAY** be used within another control structure, e.g., an "if" block.

Examples:

The user has four scripts stored in their personal repository:

"default"

This is the default active script that includes several others.

```
require ["include"];

include :personal "always_allow";
include :global "spam_tests";
include :personal "spam_tests";
include :personal "mailing_lists";
```

Personal script "always_allow"

This script special-cases some correspondent email addresses and makes sure any message containing those addresses are always kept.

```
if address :is "from" "boss@example.com"
{
    keep;
}
elsif address :is "from" "ceo@example.com"
{
    keep;
}
```


Personal script "spam_tests" (uses "reject" [RFC5429])

This script does some user-specific spam tests to catch spam messages not caught by the site-wide spam tests.

```
require ["reject"];

if header :contains "Subject" "XXXX"
{
    reject "Subject XXXX is unacceptable.";
}
elsif address :is "from" "money@example.com"
{
    reject "Mail from this sender is unwelcome.";
}
```

Personal script "mailing_lists"

This script looks for messages from different mailing lists and files each into a mailbox specific to the mailing list.

```
require ["fileinto"];

if header :is "List-ID" "sieve.ietf.org"
{
    fileinto "lists.sieve";
}
elsif header :is "List-ID" "ietf-imapext.imc.org"
{
    fileinto "lists.imapext";
}
```

There is one script stored in the global repository:

Site script "spam_tests" (uses "reject" [RFC5429])

This script does some site-wide spam tests which any user at the site can include in their own scripts at a suitable point. The script content is kept up to date by the site administrator.

```
require ["reject"];

if anyof (header :contains "Subject" "$$",
          header :contains "Subject" "Make money")
{
    reject "No thank you.";
}
```

3.3. Control Structure return

Usage: return

The "return" command stops processing of the immediately included script only and returns processing control to the script which includes it. If used in the main script (i.e., not in an included script), it has the same effect as the "stop" command, including the appropriate "keep" action if no other actions have been executed up to that point.

3.4. Interaction with Variables Extension

In order to avoid problems of variables in an included script "overwriting" those from the script that includes it, this specification requires that all variables defined in a script MUST be kept "private" to the immediate script by default - that is, they are not "visible" to other scripts. This ensures that two script authors cannot inadvertently cause problems by choosing the same name for a variable.

However, sometimes there is a need to make a variable defined in one script available to others. This specification defines the new command "global" to declare that a variable is shared among scripts. Effectively, two namespaces are defined: one local to the immediate script, and another shared among all scripts. Implementations MUST allow a non-global variable to have the same name as a global variable but have no interaction between them.

3.4.1. Control Structure global

Usage: global <value: string-list>

The "global" command accepts a string list argument that defines one or more names of variables to be stored in the global variable space. Each name MUST be a constant string and conform to the syntax of variable-name as defined in VARIABLES [RFC5229], Section 3. Match variables cannot be specified and namespace prefixes are not allowed. An invalid name MUST be detected as a syntax error.

The "global" command is only available when the script has both "include" and "variables" in its require line. If the "global" command appears when only "include" or only "variables" has been required, an error MUST be generated when the script is uploaded.

If a "global" command is given the name of a variable that has previously been defined in the immediate script with "set", an error MUST be generated either when the script is uploaded or at execution time.

If a "global" command lists a variable that has not been defined in the global namespace, the name of the variable is now marked as global, and any subsequent "set" command will set the value of the variable in global scope.

A variable has global scope in all scripts that have declared it with the "global" command. If a script uses that variable name without declaring it global, the name specifies a separate, non-global variable within that script.

Interpretation of a string containing a variable marked as global, but without any value set, SHALL behave as any other access to an unknown variable, as specified in VARIABLES [RFC5229], Section 3 (i.e., evaluates to an empty string).

Example:

The active script

The included script may contain repetitive code that is effectively a subroutine that can be factored out. In this script, the test which matches last will leave its value in the test_mailbox variable and the top-level script will file the message into that mailbox. If no tests matched, the message will be implicitly kept in the INBOX.

```
require ["fileinto", "include", "variables", "relational"];
global "test";
global "test_mailbox";

set "test" "$$";
include "subject_tests";

set "test" "Make money";
include "subject_tests";

if string :count "eq" "${test_mailbox}" "1"
{
    fileinto "${test_mailbox}";
    stop;
}
```

Personal script "subject_tests"

This script performs a number of tests against the message, sets the global test_mailbox variable with a folder to file the message into, then falls back to the top-level script.

```
require ["include", "variables"];
global ["test", "test_mailbox"];

if header :contains "Subject" "${test}"
{
    set "test_mailbox" "spam-${test}";
}
```

3.4.2. Variables Namespace global

In addition to the "global" command, this document defines the variables namespace "global", as specified in VARIABLES [RFC5229], Section 3. The global namespace has no sub-namespaces (e.g., 'set "global.data.from" "me@example.com";' is not allowed). The variable-name part MUST be a valid identifier (e.g., 'set "global.12" "value";' is not valid because "12" is not a valid identifier).

Note that VARIABLES [RFC5229], Section 3, suggests that extensions should define a namespace that is the same as its capability string (in this case, "include" rather than "global"). Nevertheless, references to the "global" namespace without a prior require statement for the "include" extension MUST cause an error.

Example:

```
require ["variables", "include"];

set "global.i_am_on_vacation" "1";
```

Variables declared global and variables accessed via the global namespace MUST each be one and the same. In the following example script, we see the variable "i_am_on_vacation" used in a "global" command, and again with the "global." namespace. Consider these as two syntaxes with identical meaning.

Example:

```
require ["variables", "include", "vacation"];
global "i_am_on_vacation";

set "global.i_am_on_vacation" "1";

if string :is "${i_am_on_vacation}" "1"
{
    vacation "It's true, I am on vacation.";
}
```

3.5. Interaction with Other Extensions

When "include" is used with the Editheader extension [RFC5293], any changes made to headers in a script MUST be propagated both to and from included scripts. By way of example, if a script deletes one header and add another, then includes a second script, the included script MUST NOT see the removed header, and MUST see the added header. Likewise, if the included script adds or removes a header, upon returning to the including script, subsequent actions MUST see the added headers and MUST NOT see the removed headers.

When "include" is used with the MIME extension [RFC5703] "foreverypart" control structure, the included script MUST be presented with the current MIME part as though it were the entire message. A script SHALL NOT have any special control over the control structure it was included from. In the MIME example once again, a "stop" or "return" in an included script cannot directly terminate or continue flow of a "foreverypart" block. In such a case, the included script should set a global variable that the including script can test. A "stop" in an included script, even within a "foreverypart" loop, still halts all script execution, per Section 3.2.

When "include" is used with the Reject extension [RFC5429], calling "reject" or "ereject" at any time sets the reject action on the message, and continues script execution. Apropos of the MIME

extension, if an included script sees only a portion of the message and calls a reject, it is the entire message and not the single MIME part which carries the rejection.

4. Security Considerations

Sieve implementations MUST ensure adequate security for the global script repository to prevent unauthorized changes to global scripts. For example, a site policy might enable only certain users with administrative privileges to modify the global scripts. Site are advised against allowing all users to have write access to the site's global scripts.

Sieve implementations MUST ensure that script names are checked for validity and proper permissions prior to inclusion, in order to prevent a malicious user from gaining access to files accessible to the mail server software that should not be accessible to the user.

Sieve implementations MUST ensure that script names are safe for use with their storage system. An error MUST be generated either when the script is uploaded or at execution time for a script including a name that could be used as a vector to attack the storage system. By way of example, the following include commands should be considered hostile: 'include "../.../etc/passwd"', 'include "foo\$(`rm star`)'".

Beyond these, the "include" extension does not raise any security considerations that are not present in the base SIEVE [RFC5228] document and the VARIABLES [RFC5229] extension.

5. IANA Considerations

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

5.1. "include" Extension Registration

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

Capability name: include
Description: adds the "include" command to execute other Sieve scripts, the "return" action from an included script, and the "global" command and "global" variables namespace to access variables shared among included scripts.
RFC number: this RFC
Contact address: the Sieve discussion list <sieve@ietf.org>

This information has been added to the IANA registry of Sieve Extensions (currently found at <http://www.iana.org>).

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5228] Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language", RFC 5228, January 2008.
- [RFC5229] Homme, K., "Sieve Email Filtering: Variables Extension", RFC 5229, January 2008.
- [RFC5804] Melnikov, A. and T. Martin, "A Protocol for Remotely Managing Sieve Scripts", RFC 5804, July 2010.

6.2. Informative References

- [RFC5293] Degener, J. and P. Guenther, "Sieve Email Filtering: Editheader Extension", RFC 5293, August 2008.
- [RFC5429] Stone, A., "Sieve Email Filtering: Reject and Extended Reject Extensions", RFC 5429, March 2009.
- [RFC5703] Hansen, T. and C. Daboo, "Sieve Email Filtering: MIME Part Tests, Iteration, Extraction, Replacement, and Enclosure", RFC 5703, October 2009.

Appendix A. Acknowledgments

Thanks to Ken Murchison, Rob Siemborski, Alexey Melnikov, Marc Mutz, Kjetil Torgrim Homme, Stephan Bosch, Arnt Gulbrandsen, Barry Leiba, and Jeffrey Hutzelman for comments and corrections.

Appendix B. Change History (to be removed prior to publication as an RFC)

Changes from ietf-14 to ietf-15:

- a. Interaction with reject, stop.

Changes from ietf-13 to ietf-14:

- a. Updated for Last Call comments.

Changes from ietf-12 to ietf-13:

- a. Nits from Stephan Bosch.
- b. Nits from Robert Burrell Donkin.

Changes from ietf-11 to ietf-12:

- a. Nits from Alexey Melnikov.
- b. Nits from Barry Leiba.

Changes from ietf-10 to ietf-11:

- a. Nits from Dilyan Palauzov.
- b. Nits from Stephan Bosch.
- c. Nits from Alexey Melnikov.

Changes from ietf-09 to ietf-10:

- a. Another example script error caught by Stephan Bosch.
- b. Add :optional argument to allow a missing script to be ignored.

Changes from ietf-08 to ietf-09:

- a. Better variables language from Stephan Bosch.

Changes from ietf-07 to ietf-08:

- a. Nits from Stephan Bosch.
- b. Nits from Barry Leiba.
- c. Wordsmithing and layout wrangling.

Changes from ietf-06 to ietf-07:

- a. Nits from Stephan Bosch.

Changes from ietf-05 to ietf-06:

- a. Nits from Barry Leiba.

Changes from ietf-04 to ietf-05:

- a. Integrate review from Barry Leiba.

Changes from ietf-03 to ietf-04:

- a. No changes.

Changes from ietf-02 to ietf-03:

- a. Setting a variable then calling global on it is an error (something like 'use strict').
- b. Specify that the 'global' keyword is only available when 'variables' has also been required.
- c. Uploading a script that includes a nonexistent script is not an error at upload time.

Changes from ietf-01 to ietf-02:

- a. Require that script names must be constant strings, not subject to variable expansion.
- b. Try the phrase immediate script instead of current script.
- c. Clarify that "global 'varname'" and "global.varname" refer to the same variable.
- d. Drop the requirement the global keywords come after require and before anything else.

Changes from ietf-00 to ietf-01:

- a. Replaced import/export with global.
- b. Added :once modifier to include.
- c. Added global namespace to see if it holds water.

Changes from daboo-06 to ietf-00:

- a. None

Changes from -05 to -06:

- a. Aaron Stone joins as author.
- b. Removed | characters from the script examples.
- c. Updated draft references to published RFCs.

Changes from -04 to -05:

- a. Fixed examples.
- b. Relaxed requirement that imported/exported variables be set before being used.

Changes from -03 to -04:

- a. Fixed missing 2119 definitions.
- b. Defined interaction with variables through use of import and export commands.

Changes from -02 to -03:

- a. Refreshing expired draft (updated for nits).
- b. Syntax -> Usage.
- c. Updated to 3028bis reference.

Changes from -01 to -02:

- a. Minor formatting changes only - refreshing expired draft.

Changes from -00 to -01:

- a. Added IPR boiler plate.
- b. Re-ordered sections at start to conform to RFC style.
- c. Moved recursion comment into General Considerations section.
- d. Switched to using optional parameter to indicate personal vs global.
- e. Explicitly state that an error occurs when a missing script is included.

Authors' Addresses

Cyrus Daboo

Email: cyrus@daboo.name

Aaron Stone

Email: aaron@serendipity.cx

Sieve Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 13, 2012

A. Melnikov
Isode Limited
Q. Sun
B. Leiba
K. Li
Huawei Technologies
October 11, 2011

Sieve Notification Mechanism: SIP MESSAGE
draft-ietf-sieve-notify-sip-message-08

Abstract

This document describes a profile of the Sieve extension for notifications, to allow notifications to be sent over SIP 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 April 13, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Overview	3
1.2.	Terminology	3
2.	Definition	3
2.1.	Notify parameter "method"	3
2.2.	Notify tag ":from"	4
2.3.	Notify tag ":options"	4
2.4.	Notify tag ":importance"	4
2.5.	Notify tag ":message"	5
2.6.	Other Definitions	5
2.7.	Test notify_method_capability	5
3.	Examples	6
3.1.	Example 1	6
3.2.	Example 2	6
4.	Requirements Conformance Checklist	7
5.	Security Considerations	8
6.	IANA Considerations	9
7.	Acknowledgements	9
8.	References	10
8.1.	Normative References	10
8.2.	Informative References	10
	Authors' Addresses	10

1. Introduction

1.1. Overview

The Notify extension [RFC5435] to the Sieve mail filtering language [RFC5228] is a framework for providing notifications by employing URIs that specify the notification mechanism. (See RFC 5435 for details about the motivation and use cases.) This document defines how Session Initiation Protocol (SIP) URIs RFC 3261 [RFC3261] are used to generate notifications via SIP MESSAGE RFC 3428 [RFC3428].

1.2. Terminology

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 [RFC2119].

This document inherits terminology from the Sieve email filtering language [RFC5228], the Sieve Notify extension [RFC5435], and RFC 3261 [RFC3261].

2. Definition

The SIP MESSAGE mechanism defined in this document results in the sending of a SIP MESSAGE request to notify a recipient about an email message.

2.1. Notify parameter "method"

The "method" parameter MUST be a URI that conforms to the SIP or SIPS URI scheme (as specified in RFC 3261 [RFC3261]) and that identifies a SIP or SIPS recipient of the notification. The URI MAY include the resource identifier portion of a SIP address and URI parameters. The URI MUST include the URI parameter "method", with the value "MESSAGE". Example:

```
notify "sip:romeo@example.com;method=MESSAGE"
```

Note that future specifications might extend this document and define Sieve notifications that use SIP methods other than "MESSAGE".

The processing application MUST form a request according to the rules specified in RFC 3261 [RFC3261].

Note that other URI schemes can also trigger SIP processing, but only SIP and SIPS are defined here. Future extensions might define other

Sieve notification methods that use SIP through other URI schemes.

2.2. Notify tag ":from"

The value of the ":from" tag MUST use the SIP "From" header field syntax; if the ":from" value is specified, has valid syntax, and is valid according to the implementation-specific security checks (see Section 3.3 of Sieve Notify [RFC5435]), then the notification SHOULD include the "From" SIP header field containing the value of the ":from" notify tag. If the specified value is not valid, then it is ignored.

All SIP authentication, including challenges and client certificates, SHOULD be done in the context of the Sieve engine -- the Sieve engine is the identity being authenticated. This avoids security issues associated with the Sieve engine's having access to the end user's SIP authentication credentials. The Sieve engine MAY use server-wide credentials (including applicable certificates) that are the same for all scripts. Alternatively, it MAY, for auditing purposes, use different sets of Sieve-engine credentials when operating on behalf of different users.

See section 22 of RFC 3261 [RFC3261] for more information about SIP authentication.

2.3. Notify tag ":options"

Handling of the ":options" tag is implementation specific. This document doesn't require presence of any option and doesn't define how options are processed.

2.4. Notify tag ":importance"

The ":importance" tag is intended to convey the importance of the SIP MESSAGE notification, not the importance of the email message that generated the notification. The value of the ":importance" tag MAY, therefore, be transformed into SIP "Priority" header field (in addition to or instead of including it in the body of the message). Note that because the Sieve ":importance" tag only has three values, not all SIP "Priority" values can be represented in the transformation. If this transformation is done, the value of the "Priority" header field MUST be "urgent" if the value of the ":importance" tag is "1", "normal" if the value of the ":importance" tag is "2", and "non-urgent" if the value of the ":importance" tag is "3". There is no mapping to the SIP value "emergency", nor to any additional values that might be defined.

2.5. Notify tag ":message"

If the ":message" tag is included, it MUST be transformed into the message-body of a SIP MESSAGE, which MUST have Content-Type value of "text/plain" with CHARSET="UTF-8". If the ":message" tag is not included, a default message will be used. The values of the "From" and "Subject" header fields of the triggering email message are particularly useful to users receiving notifications, and including them in the default message is generally a good idea (but see the Security Considerations, Section 5). The default body might also include the value of the ":importance" tag, if one is specified), as shown in Section 3.2 below.

Note that in no case is the actual triggering message body included in the notification.

Implementations MUST comply with the SIP MESSAGE size limits, as discussed in section 8 of RFC 3428 [RFC3428].

2.6. Other Definitions

An implementation MUST ignore any URI parameter it does not understand (the URI MUST be processed as if the parameter were not present). The URI "body" parameter can serve the same purpose as the Sieve ":message" tag, providing the message body of the SIP MESSAGE request. If both are present at the same time, the Sieve processing MUST ignore the "body" parameter.

Using the ":message" tag has advantages over using the "body" parameter. Because the ":message" tag is part of the "notify" statement syntax, it can be easier to include it in a script, and to do things such as variable substitutions [RFC5229] with it. It is also easier to include non-ASCII characters in the ":message" tag, because such characters have to be encoded if they are within URI parameters, but can be included directly in UTF-8 in Sieve tag values.

The policy for retrying delivery of failed notifications is specified in RFC 3261 [RFC3261], according to the SIP error code returned during an attempt to deliver a SIP notification. In other words, unlike the situation with some other Sieve notification methods, retries for SIP MESSAGE notifications are controlled by the notification protocol itself (SIP).

2.7. Test notify_method_capability

Absent use of SIP extensions such as [RFC3856], it is impossible to tell in advance whether the notification recipient is online and able

to receive a SIP MESSAGE. Expect the `notify_method_capability` test for "online" to frequently return "maybe" for this notification method.

3. Examples

In the following examples, the sender of the email has an address of `juliet@example.org`, the entity to be notified has a SIP address of `<sip:romeo@example.com>`, and the notification service has a SIP address `<sip:notifier@example.com>`.

3.1. Example 1

The following is a basic Sieve notify action with only a method:

```
notify "sip:romeo@example.com;method=MESSAGE"
```

The resulting SIP MESSAGE request might be as follows:

```
MESSAGE sip:romeo@example.com SIP/2.0
Via: SIP/2.0/TCP notifier.example.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
From: sip:notifier@example.com;tag=32328
To: sip:romeo@example.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Date: Sat, 13 Nov 2010 23:29:00 GMT
Content-Type: text/plain
Content-Length: 53
```

```
<juliet@example.com> wrote: Contact me immediately!
```

In the example above the email message was received from `juliet@example.com` and had "Subject: Contact me immediately!"

3.2. Example 2

The following is a more advanced Sieve notify action with a method, importance, subject, and message:

```
notify :importance "1"
      :message "You got new mail!"
      "sip:romeo@example.com;method=MESSAGE?subject=SIEVE"

MESSAGE sip:romeo@example.com SIP/2.0
Via: SIP/2.0/TCP notifier.example.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
From: sip:notifier@example.com;tag=32328
To: sip:romeo@example.com
Subject: SIEVE
Priority: urgent
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Date: Fri, 08 Apr 2011 06:54:00 GMT
Content-Type: text/plain
Content-Length: 19

You got new mail!
```

4. Requirements Conformance Checklist

Section 3.8 of Sieve Notify [RFC5435] specifies a set of requirements for Sieve notification methods. A checklist is provided here to show conformance of the SIP MESSAGE notification method.

1. No new Sieve tags have been added to the "notify" action.
2. An implementation of the SIP MESSAGE notification method SHOULD NOT modify the final notification text, except to comply with SIP MESSAGE length limits. Deployments MAY make operational decisions about notification text, for reasons such as privacy and confidentiality. Modification of characters themselves should not be necessary, since the SIP MESSAGE body is encoded in UTF-8 [RFC3629].
3. An implementation MAY ignore parameters specified in the ":importance", and ":options" tags.
4. A default message is suggested in Section 2.5.
5. A notification sent via the SIP MESSAGE notification method MAY include the Date header field containing the date-time of the moment when the SIP MESSAGE notification was generated.
6. The notification source is identified through the SIP "From:" header field, via the Sieve Notify ":from" tag (see Section 2.2).

7. An implementation **MUST NOT** include any other extraneous information not specified in parameters to the notify action.
8. An implementation **MUST** ignore any URI parameters it does not understand (i.e., the URI **MUST** be processed as if the action or parameter were not present). See Section 2.6 for more details.
9. The notify_method_capability test for the "online" notification-capability behaves as described in Section 2.7.
10. The policy for retrying delivery of failed notifications is specified in RFC 3261 [RFC3261], as noted in Section 2.6.

5. Security Considerations

Depending on the information included, sending a notification can be, from a confidentiality point of view, comparable to forwarding mail to the notification recipient. Care must be taken when automatically forwarding information such as the sender and the subject of a message, to ensure that confidential information is not sent into an insecure environment or over an insecure channel. Depending upon the environment, this might entail using SIPS URIs, not sending information about the subject and/or the sender, or applying heuristics to the message to determine what may be sent.

As required by RFC 3428, user agents that support the SIP MESSAGE request **MUST** implement end-to-end authentication, body integrity, and body confidentiality mechanisms. At the time of this writing, there is not widespread deployment of SIP end-to-end security, so there can be cases where it is not possible to use it, even though it is implemented on one end. It's important to note that such situations are open to exposure of user credentials, message content, and other private information via man-in-the-middle and other passive attacks.

The Sieve Notify extension specifies that notification methods **MUST** provide mechanisms for avoiding notification loops. In this case, the SIP protocol itself prevents loops, and no explicit work is needed within the notification mechanism. In situations where a SIP MESSAGE notification can result in an email message, which could generate another SIP MESSAGE notification, loop prevention through rate detection and limiting might be necessary. An implementation might detect too many notifications within a given time period, too many triggered by a particular sender, too many with the same subject, or the like, and shut off the affected notifications for a period of time or until manual intervention turns them back on.

If SIP MESSAGE requests might be billed by the message, or the use of

them might deplete a user's quota of messages, notification by this mechanism can present a situation where someone using a large number of messages to generate a large number of notifications will cause a significant expense to the recipient. Because there is no external way an attacker can tell that this is the case, such an attack would likely be a random or nuisance attack. Nevertheless, users might be warned of potential costs when they set up SIP MESSAGE notifications.

Other security considerations given in the Sieve base specification [RFC5228], the Sieve Notify extension [RFC5435], and RFC 3261 [RFC3261] are also relevant to this document.

6. IANA Considerations

The following template provides the IANA registration of the Sieve notification mechanism specified in this document. This information should be added to the list of Sieve notification mechanisms maintained at <http://www.iana.org/assignments/sieve-notification>.

To: iana@iana.org
Subject: Registration of new Sieve notification mechanism
Mechanism name: sip-message
Mechanism URI: SIP/SIPS as specified in RFC 3261 [RFC3261]
Mechanism-specific options: none
Standards Track/IESG-approved experimental RFC number: [RFC XXXX]
Person and email address to contact for further information:
 See authors of [RFC XXXX]

7. Acknowledgements

This document borrows some text from draft-ietf-sieve-notify-xmpp [RFC5437].

Henning Schulzrinne (hgs@cs.columbia.edu) was a special contributor to this document, with early work and reviews.

The authors would like to thank Adam Roach and Eric Burger for their helpful comments. Ben Campbell did a very thorough RAI-team review, as well as a follow-up review to make sure we resolved all of his issues satisfactorily. This document was greatly improved by their input.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC5228] Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language", RFC 5228, January 2008.
- [RFC5435] Melnikov, A., Leiba, B., Segmuller, W., and T. Martin, "Sieve Email Filtering: Extension for Notifications", RFC 5435, January 2009.

8.2. Informative References

- [RFC3856] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.
- [RFC5229] Homme, K., "Sieve Email Filtering: Variables Extension", RFC 5229, January 2008.
- [RFC5437] Saint-Andre, P. and A. Melnikov, "Sieve Notification Mechanism: Extensible Messaging and Presence Protocol (XMPP)", RFC 5437, January 2009.

Authors' Addresses

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com
URI: <http://www.melnikov.ca/>

Qian Sun
Huawei Technologies
Bantian, Longgang
Shenzhen, Guandong 518129
P.R China

Phone: +86 755 28780808
Email: sunqian@huawei.com

Barry Leiba
Huawei Technologies

Phone: +1 646 827 0648
Email: barryleiba@computer.org
URI: <http://internetmessagingtechnology.org/>

Kepeng Li
Huawei Technologies
Huawei Base, Bantian, Longgang District
Shenzhen, Guangdong 518129
P. R. China

Phone: +86-755-28974289
Email: likepeng@huawei.com

