Internet Engineering Task Force (IETF)
Request for Comments: 6558
Category: Standards Track
ISSN: 2070-1721

A. Melnikov
Isode Limited
B. Leiba
K. Li
Huawei Technologies

March 2012

Sieve Extension for Converting Messages before Delivery

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 is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6558.

Table of Contents

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.

2. "convert" Action

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

The "convert" action specifies that all body parts with a media type [RFC2046] (sometimes called "MIME type") equal to <quoted-from-media-type> be converted to the media type in <quoted-to-media-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-media-type> media 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-media-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 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; 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 or not the actual conversion has been done yet, 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 has added 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: RFC 6558
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.

Qian Sun contributed text to this document.

7. Normative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[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.

[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/

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