Network Working Group
Request for Comments: 5173
Updates: 5229
Category: Standards Track

J. Degener
P. Guenther
Sendmail, Inc.

April 2008

Sieve Email Filtering: Body Extension

Status of This Memo

Abstract

This document defines a new command for the "Sieve" email filtering language that tests for the occurrence of one or more strings in the body of an email message.

1. Introduction

The "body" test checks for the occurrence of one or more strings in the body of an email message. Such a test was initially discussed for the [SIEVE] base document, but was subsequently removed because it was thought to be too costly to implement.

Nevertheless, several server vendors have implemented some form of the "body" test.

This document reintroduces the "body" test as an extension, and specifies its syntax and semantics.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEYWORDS].

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

The rules for interpreting the grammar are defined in [SIEVE] and inherited by this specification. In particular, readers of this document are reminded that according to [SIEVE] Sections 2.6.2 and 2.6.3, optional arguments such as COMPARATOR and MATCH-TYPE can appear in any order.

3. Capability Identifier

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

4. Test body

Usage: "body" [COMPARATOR] [MATCH-TYPE] [BODY-TRANSFORM]
<key-list: string-list>

The body test matches content in the body of an email message, that is, anything following the first empty line after the header. (The empty line itself, if present, is not considered to be part of the body.)

The COMPARATOR and MATCH-TYPE keyword parameters are defined in [SIEVE]. As specified in Sections 2.7.1 and 2.7.3 of [SIEVE], the default COMPARATOR is "i;ascii-casemap" and the default MATCH-TYPE is ":is".

The BODY-TRANSFORM is a keyword parameter that governs how a set of strings to be matched against are extracted from the body of the message. If a message consists of a header only, not followed by an empty line, then that set is empty and all "body" tests return false, including those that test for an empty string. (This is similar to how the "header" test always fails when the named header fields aren't present.) Otherwise, the transform must be followed as defined below in Section 5.

Note that the transformations defined here do *not* match against each line of the message independently, so the strings will usually contain CRLFs. How these can be matched is governed by the comparator and match-type. For example, with the default comparator of "i;ascii-casemap", they can be included literally in the key strings, or be matched with the "*" or "?" wildcards of the :matches match-type, or be skipped with :contains.

5. Body Transform

Prior to matching content in a message body, "transformations" can be applied that filter and decode certain parts of the body. These transformations are selected by a "BODY-TRANSFORM" keyword parameter.

```
Usage: ":raw"
     / ":content" <content-types: string-list>
     / ":text"
```

The default transformation is :text.

5.1. Body Transform ":raw"

The ":raw" transform matches against the entire undecoded body of a message as a single item.

If the specified body-transform is ":raw", the [MIME] structure of the body is irrelevant. The implementation MUST NOT remove any transfer encoding from the message, MUST NOT refuse to filter messages with syntactic errors (unless the environment it is part of rejects them outright), and MUST treat multipart boundaries or the MIME headers of enclosed body parts as part of the content being matched against, instead of MIME structures to interpret.

Example:

```
require "body";

# This will match a message containing the literal text
# "MAKE MONEY FAST" in body parts (ignoring any
# content-transfer-encodings) or MIME headers other than
# the outermost RFC 2822 header.

if body :raw :contains "MAKE MONEY FAST" {
        discard;
}
```

5.2. Body Transform ":content"

If the body transform is ":content", the MIME parts that have the specified content types are matched against independently.

If an individual content type begins or ends with a '/' (slash) or contains multiple slashes, then it matches no content types. Otherwise, if it contains a slash, then it specifies a full <type>/<subtype> pair, and matches only that specific content type. If it is the empty string, all MIME content types are matched. Otherwise, it specifies a <type> only, and any subtype of that type matches it.

The search for MIME parts matching the :content specification is recursive and automatically descends into multipart and message/rfc822 MIME parts. All MIME parts with matching types are searched for the key strings. The test returns true if any combination of a searched MIME part and key-list argument match.

If the :content specification matches a multipart MIME part, only the prologue and epilogue sections of the part will be searched for the key strings, treating the entire prologue and the entire epilogue as separate strings; the contents of nested parts are only searched if their respective types match the :content specification.

If the :content specification matches a message/rfc822 MIME part, only the header of the nested message will be searched for the key strings, treating the header as a single string; the contents of the nested message body parts are only searched if their content type matches the :content specification.

For other MIME types, the entire part will be searched as a single string.

(Matches against container types with an empty match string can be useful as tests for the existence of such parts.)

Example:

```
     From: Whomever
     To: Someone
     Date: Whenever
     Subject: whatever
     Content-Type: multipart/mixed; boundary=outer

   &  This is a multi-part message in MIME format.
   &
     --outer
     Content-Type: multipart/alternative; boundary=inner

   &  This is a nested multi-part message in MIME format.
   &
     --inner
     Content-Type: text/plain; charset="us-ascii"

   $  Hello
   $
     --inner
     Content-Type: text/html; charset="us-ascii"

   %  <html><body>Hello</body></html>
   %
     --inner--
   &
   &  This is the end of the inner MIME multipart.
   &
     --outer
     Content-Type: message/rfc822

   !  From: Someone Else
   !  Subject: hello request

   $  Please say Hello
   $
     --outer--
   &
   &  This is the end of the outer MIME multipart.
```

In the above example, the '&', '$', '%', and '!' characters at the start of a line are used to illustrate what portions of the example message are used in tests:

- the lines starting with '&' are the ones that are tested when a 'body :content "multipart" :contains "MIME"' test is executed.

- the lines starting with '$' are the ones that are tested when a 'body :content "text/plain" :contains "Hello"' test is executed.

- the lines starting with '%' are the ones that are tested when a 'body :content "text/html" :contains "Hello"' test is executed.

- the lines starting with '$' or '%' are the ones that are tested when a 'body :content "text" :contains "Hello"' test is executed.

- the lines starting with '!' are the ones that are tested when a 'body :content "message/rfc822" :contains "Hello"' test is executed.

Comparisons are performed on octets. Implementations decode the content-transfer-encoding and convert text to [UTF-8] as input to the comparator. MIME parts that cannot be decoded and converted MAY be treated as plain US-ASCII, omitted, or processed according to local conventions. A NUL octet (character zero) SHOULD NOT cause early termination of the content being compared against. Implementations MUST support the "quoted-printable", "base64", "7bit", "8bit", and "binary" content transfer encodings. Implementations MUST be capable of converting to UTF-8 the US-ASCII, ISO-8859-1, and the US-ASCII subset of ISO-8859-* character sets.

Each matched part is matched against independently: search expressions MUST NOT match across MIME part boundaries. MIME headers of the containing part MUST NOT be included in the data.

Example:

```
require ["body", "fileinto"];

# Save any message with any text MIME part that contains the
# words "missile" or "coordinates" in the "secrets" folder.

if body :content "text" :contains ["missile", "coordinates"] {
        fileinto "secrets";
}

# Save any message with an audio/mp3 MIME part in
# the "jukebox" folder.

if body :content "audio/mp3" :contains "" {
        fileinto "jukebox";
}
```

5.3. Body Transform ":text"

The ":text" body transform matches against the results of an implementation's best effort at extracting UTF-8 encoded text from a message.

It is unspecified whether this transformation results in a single string or multiple strings being matched against. All the text extracted from a given non-container MIME part MUST be in the same string.

In simple implementations, :text MAY be treated the same as :content "text".

Sophisticated implementations MAY strip mark-up from the text prior to matching, and MAY convert media types other than text to text prior to matching.

(For example, they may be able to convert proprietary text editor formats to text or apply optical character recognition algorithms to image data.)

Example:

```
require ["body", "fileinto"];

# Save messages mentioning the project schedule in the
# project/schedule folder.
if body :text :contains "project schedule" {
        fileinto "project/schedule";
}
```

6. Interaction with Other Sieve Extensions

Any extension that extends the grammar for the COMPARATOR or MATCH-TYPE nonterminals will also affect the implementation of "body".

Wildcard expressions used with "body" are exempt from the side effects described in [VARIABLES]. That is, they MUST NOT set match variables (${1}, ${2}...) to the input values corresponding to wildcard sequences in the matched pattern. However, if the extension is present, variable references in the key strings or content type strings are evaluated as described in this document.

7. IANA Considerations

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

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

Capability name: body
Description: Provides a test for matching against the body of the message being processed
RFC number: RFC 5173
Contact Address: The Sieve discussion list <ietf-mta-filters@imc.org>

8. Security Considerations

The system MUST be sized and restricted in such a manner that even malicious use of body matching does not deny service to other users of the host system.

Filters relying on string matches in the raw body of an email message may be more general than intended. Text matches are no replacement for a spam, virus, or other security related filtering system.

9. Acknowledgments

This document has been revised in part based on comments and discussions that took place on and off the SIEVE mailing list. Thanks to Cyrus Daboo, Ned Freed, Bob Johannessen, Simon Josefsson, Mark E. Mallett, Chris Markle, Alexey Melnikov, Ken Murchison, Greg Shapiro, Tim Showalter, Nigel Swinson, Dowson Tong, and Christian Vogt for reviews and suggestions.

10. References

10.1. Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

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

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

10.2. Informative References

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

Authors' Addresses

Jutta Degener
5245 College Ave, Suite #127
Oakland, CA 94618

EMail: jutta@pobox.com

Philip Guenther
Sendmail, Inc.
6425 Christie Ave, 4th Floor
Emeryville, CA 94608

EMail: guenther@sendmail.com

Full Copyright Statement

Intellectual Property