

Internet Engineering Task Force	T. Hansen	
Internet-Draft	AT&T Laboratories	
Intended status: Standards Track	C. Daboo	
Expires: January 10, 2008	Apple Computer	
	July 09, 2007	

[TOC](#)

**Sieve Email Filtering: MIME part Tests, Iteration, Extraction, Replacement and Enclosure  
draft-ietf-sieve-mime-loop-03**

**Status of this Memo**

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

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

This Internet-Draft will expire on January 10, 2008.

**Abstract**

The Sieve email filtering language has no way to examine individual MIME parts or any way to manipulate those individual parts. However, being able to filter based on MIME content is important. This document defines extensions for these needs.

**Note**

This document is being discussed on the MTA-FILTERS mailing list, [ietf-mta-filters@imc.org](mailto:ietf-mta-filters@imc.org).

---

## Table of Contents

<a href="#">1.</a>	Introduction
<a href="#">2.</a>	Conventions Used in This Document
<a href="#">3.</a>	Sieve Loops
<a href="#">4.</a>	Changes to Sieve tests
<a href="#">4.1.</a>	Test "header"
<a href="#">4.2.</a>	Test "address"
<a href="#">4.3.</a>	Test "exists"
<a href="#">5.</a>	Action Replace
<a href="#">6.</a>	Action Enclose
<a href="#">7.</a>	Action extract_text
<a href="#">8.</a>	Sieve Capability Strings
<a href="#">9.</a>	Examples
<a href="#">9.1.</a>	Example 1
<a href="#">9.2.</a>	Example 2
<a href="#">9.3.</a>	Example 3
<a href="#">10.</a>	Acknowledgements
<a href="#">11.</a>	Security Considerations
<a href="#">12.</a>	IANA Considerations
<a href="#">13.</a>	Change History (to be removed prior to publication as an RFC)
<a href="#">13.1.</a>	draft-ietf-sieve-mime-02
<a href="#">13.2.</a>	draft-ietf-sieve-mime-01
<a href="#">13.3.</a>	draft-ietf-sieve-mime-00
<a href="#">13.4.</a>	draft-hansen-sieve-loop-01
<a href="#">13.5.</a>	draft-hansen-sieve-loop-02
<a href="#">13.6.</a>	draft-hansen-sieve-loop-03
<a href="#">14.</a>	References
<a href="#">14.1.</a>	Normative References
<a href="#">14.2.</a>	Informative References
<a href="#">§</a>	Authors' Addresses
<a href="#">§</a>	Intellectual Property and Copyright Statements

---

## 1. Introduction

[TOC](#)

Sieve scripts are used to make decisions about the disposition of an email message. The base Sieve specification, [\[I-D.ietf-sieve-3028bis\]](#) (Showalter, T. and P. Guenther, "Sieve: An Email Filtering Language," October 2007.), defines operators for looking at the message headers, such as addresses and the subject. Other extensions provide access to the body of the message ([\[I-D.ietf-sieve-body\]](#) (Guenther, P. and J. Degener, "Sieve Email Filtering: Body Extension," March 2008.)), or allow you to manipulate the header of the message ([\[I-D.ietf-sieve-editheader\]](#) (Guenther, P. and J. Degener, "Sieve Email Filtering: Editheader Extension," March 2008.)). But none of these

extensions take into account that MIME messages ([\[RFC2045\] \(Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions \(MIME\) Part One: Format of Internet Message Bodies," November 1996.\)](#)) are often complex objects, consisting of many parts and sub-parts. This extension defines mechanisms for performing tests on MIME body parts, looping through the MIME body parts, extracting information from a MIME body part, changing the contents of a MIME body part, and enclosing the entire message with a wrapper.

---

## 2. Conventions Used in This Document

[TOC](#)

Conventions for notations are as in [\[I-D.ietf-sieve-3028bis\] \(Showalter, T. and P. Guenther, "Sieve: An Email Filtering Language," October 2007.\)](#) section 1.1.

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 [\[RFC2119\] \(Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.\)](#).

---

## 3. Sieve Loops

[TOC](#)

The base Sieve language has no looping mechanism. Given that messages may contain multiple parts, in order to support filters that apply to any and all parts, we introduce a new control command:

"for\_every\_part", which is an iterator that walks through every MIME part of a message, including nested parts, and applies the commands in the specified block to each of them. The iterator will start with the first MIME part (as its current context) and will execute a command block (Sieve commands enclosed by { ...}). Upon completion of this command block, the iterator advances to the next MIME part (as its current context) and executes the same command block again.

The iterator can be terminated prematurely by a new Sieve command, "break".

**Usage:** for\_every\_part block

**Usage:** break;

"for\_every\_part" commands can be nested inside other "for\_every\_part" commands. When this occurs, the nested "for\_every\_part" iterates over the MIME parts contained within the MIME part current being targeted by the nearest enclosing "for\_every\_part" command. If that MIME part is a

terminal MIME part (i.e. does not contain other MIME parts) then the nested "for\_every\_loop" is simply ignored.  
Sieve implementations MAY limit the number of nested loops that occur within one another, however they MUST support at least one nested loop inside another loop.

---

## 4. Changes to Sieve tests

[TOC](#)

This specification extends the base Sieve "header", "address" and "exists" tests to support targeting those tests at a specific MIME part or at all MIME parts in the enclosing scope.

---

### 4.1. Test "header"

[TOC](#)

The "header" test is extended with the addition of a new ":mime" tagged argument, which takes a number of other arguments.

```
Usage: header [:mime] [:anychild] [MIMEOPTS] [COMPARATOR] [MATCH-  
TYPE]  
      <header-names: string-list> <key-list: string-list>
```

**Usage:** The definition of [MIMEOPTS] is:

```
Syntax: ":type" / ":subtype" / ":contenttype" / ":param" <param-  
list: string-list>
```

When the ":mime" tagged argument is present in the "header" test, it will parse the MIME header lines in a message so that tests can be performed on specific elements.

If the ":anychild" tagged argument is NOT specified:

- \*If used within the context of a "for\_every\_part" iterator, the "header" test will examine the headers associated with the current MIME part context from the loop.

- \*If used outside the context of a "for\_every\_part" iterator, the "header" test will examine only the outer, top-level, headers of the message.

If the ":anychild" tagged argument IS specified, the "header" test will examine all MIME body parts and return true if any of them satisfies the test.

The "header" test with the ":mime" tagged argument can test various aspects of certain structured MIME headers. These options are available:

**:type**

parses the header assuming it has the format of a "Content-Type:" MIME header field, and tests the value of the MIME type specified in the header.

**:subtype**

parses the header assuming it has the format of a "Content-Type:" MIME header field, and tests the value of the MIME subtype specified in the header.

**:contenttype**

parses the header assuming it has the format of a "Content-Type:" MIME header field, and tests the combined value of the MIME type and subtype specified in the header.

**:param**

parses the header looking for MIME parameters in the header. The supplied string-list lists the names of any parameters to be tested. If any one named parameter value matches the test string value, the test will return true.

Example:

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

if header :mime :type "Content-Type" "image"
{
    fileinto "INBOX.images";
}
```

In this example, any message that contains a MIME image type part at the top-level is saved to the mailbox "INBOX.images".

Example:

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

if header :mime :anychild :contenttype :comparator
    "Content-Type" "text/html"
{
    fileinto "INBOX.html";
}
```

In this example, any message that contains any MIME part with a content-type of "text/html" is saved to the mailbox "INBOX.html".

Example:

```

require ["mime", "for_every_part", "fileinto"];

for_every_part
{
  if header :mime :param "filename" :comparator
    "Content-Disposition" "important"
  {
    fileinto "INBOX.important";
    break;
  }
}

```

In this example, any message that contains any MIME part with a content-disposition with a filename parameter containing the text "important" is saved to the mailbox "INBOX.important".

---

## 4.2. Test "address"

[TOC](#)

The "address" test is extended with the addition of a new ":mime" tagged argument, which takes a number of other arguments.

**Usage:** address [:mime] [:anychild] [COMPARATOR] [ADDRESS-PART]  
 [MATCH-TYPE]  
 <header-list: string-list> <key-list: string-list>

When the ":mime" tagged argument is present in the "address" test, it will parse the MIME header lines as if they were standard address header lines in a message so that tests can be performed on specific elements.

The behavior of the ":anychild" tagged argument and the interaction with the "for\_every\_part" iterator is the same as for the extended "header" test [Section 4.1 \(Test "header"\)](#).

Example:

```

require ["mime", "fileinto"];

if address :mime :is :all "content-from" "tim@example.com"
{
  fileinto "INBOX.part-from-tim";
}

```

In this example, any message that contains a MIME Content-From header at the top-level matching the text "tim@example.com" is saved to the mailbox "INBOX.part-from-time".

---

### 4.3. Test "exists"

[TOC](#)

The "exists" test is extended with the addition of a new ":mime" tagged argument, which takes one other argument.

**Usage:** exists [:mime] [:anychild] <header-names: string-list>

When the ":mime" tagged argument is present in the "exists" test, the test is extended to check for the existence of MIME headers in MIME parts.

The behavior of the ":anychild" tagged argument and the interaction with the "for\_every\_part" iterator is the same as for the extended "header" test [Section 4.1 \(Test "header"\)](#).

Example:

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

if exists :mime :anychild "content-md5"
{
    fileinto "INBOX.md5";
}
```

In this example, any message that contains a MIME Content-MD5 header in any MIME part is saved to the mailbox "INBOX.md5".

---

## 5. Action Replace

[TOC](#)

**Usage:** replace [:mime] [:subject string] [:from string]  
<replacement: string>

The "replace" command is defined to allow a MIME part to be replaced with the text supplied in the command.

When used in the context of a "for\_every\_part" iterator, the MIME part to be replaced is the "current" MIME part. If the current MIME context is a multipart MIME part, the entire multipart MIME part is replaced, which would alter the MIME structure of the message by eliminating all of the children of the multipart part. (Replacing a non-multipart MIME part within a "for\_every\_part" loop context does not alter the overall message structure.) If the MIME structure is altered, the change takes effect immediately: the "for\_every\_part" iterator that is executing does not go into the no-longer existing body parts, and subsequent "for\_every\_part" iterators would use the new message structure.

When used outside the context of a "for\_every\_part" loop, the MIME part to be replaced is the entire message.

If the :mime parameter is not specified, the replacement string is a text/plain part.

If the `:mime` parameter is specified, then the replacement string is, in fact, a MIME entity as defined in [\[RFC2045\] \(Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions \(MIME\) Part One: Format of Internet Message Bodies," November 1996.\)](#) section 2.4, including both MIME headers and content. If the optional `:mime` parameter is not supplied, the reason string is considered to be a UTF-8 string.

If the entire message is being replaced, a `:subject` parameter specifies a subject line to attach to the message that is generated. UTF-8 characters can be used in the string argument; implementations MUST convert the string to [\[RFC2047\] \(Moore, K., "MIME \(Multipurpose Internet Mail Extensions\) Part Three: Message Header Extensions for Non-ASCII Text," November 1996.\)](#) encoded words if and only if non-ASCII characters are present. Implementations MUST preserve the previous Subject header as an Original-Subject header.

If the entire message is being replaced, a `:from` parameter may be used to specify an alternate address to use in the From field of the message that is generated. The string must specify a valid [\[RFC2822\] \(Resnick, P., "Internet Message Format," April 2001.\)](#) mailbox-list. Implementations SHOULD check the syntax and generate an error when a syntactically invalid `:from` parameter is specified. Implementations MAY also impose restrictions on what addresses can be specified in a `:from` parameter; it is suggested that values that fail such a validity check simply be ignored rather than causing the replace action to fail. Implementations MUST preserve the previous From header as an Original-From header.

---

## 6. Action Enclose

[TOC](#)

**Usage:** `enclose <:subject string> <:headers string-list> string`

A new Sieve action command is defined to allow an entire message to be enclosed as an attachment to a new message. After enclosure, subsequent actions affecting the message header or content use the newly create message instead of the original message; this means that any use of a "replace" action or other similar actions should be executed before the "enclose" action.

If multiple "enclose" actions are executed by a script, only the text specified on the last one is used when creating the enclosed message. This action does not affect messages that are forwarded via a "redirect" action.

Specifically, the original message becomes a multipart/mixed message with two parts: a text/plain portion with the string argument as its body, and a message/rfc822 portion with the original message enclosed. The Content-Type: header field becomes multipart/mixed. The Subject: header is specified by the `:subject` argument. Any headers specified by



:headers are copied from the old message into the new message. If not specified by :headers, Date: and From: headers should be synthesized to reflect the current date and the user running the Sieve action.

---

## 7. Action extract\_text

[TOC](#)

**Usage:** `extract_text [MODIFIER] [":first" number] <varname: string>`

The `extract_text` action may be used within the context of a "for\_every\_part" loop. It stores at most :first bytes of the current MIME body part in the variable identified by varname. If the :first parameter is not present, the whole content of the current MIME body part is stored. In either case the actually stored data MAY be truncated to conform to implementation specific limit on variable length and/or on MIME body part length. QUESTION: What do we do if the Content-Transfer-Encoding is anything other than 7bit? If `extract_text` is used outside the context of a "for\_every\_part" loop, the action will set the variable identified by varname to the empty string.

Modifiers are applied on the extracted text before it is stored in the variable. See [\[I-D.ietf-sieve-variables\] \(Homme, K., "Sieve Extension: Variables," December 2005.\)](#) for details.

---

## 8. Sieve Capability Strings

[TOC](#)

A Sieve implementation that defines the "for\_every\_part" and "break" actions will advertise the capability string "for\_every\_part".

A Sieve implementation that defines the ":mime" tagged arguments to the "header", "address" and "exists" commands will advertise the capability string "mime".

A Sieve implementation that defines the "replace" action will advertise the capability string "replace".

A Sieve implementation that defines the "enclose" action will advertise the capability string "enclose".

A Sieve implementation that defines the "extract\_text" action will advertise the capability string "extract\_text". Note that to be useful, the "extract\_text" action also requires the "variables" [\[I-D.ietf-sieve-variables\] \(Homme, K., "Sieve Extension: Variables," December 2005.\)](#) and "mime" capabilities.

---

[TOC](#)

## 9. Examples

---

### 9.1. Example 1

[TOC](#)

A Sieve script to replace all the Windows executable attachments in a message would be:

```
require [ "for_every_part", "mime", "replace" ];
for_every_part
{
  if ( anyof (
    header :mime :contenttype :is "Content-Type" "application/exe",
    header :mime :param "filename"
      ["Content-Type", "Content-Disposition"] :matches "*.com" )
  {
    replace "Executable attachment removed by user filter";
  }
}
```

---

### 9.2. Example 2

[TOC](#)

A Sieve script to warn the user about executable attachment types would be:

```

require [ "for_every_part", "mime", "enclose" ];

for_every_part
{
    if header :mime :param "filename"
        ["Content-Type", "Content-Disposition"] :matches
            ["*.com", "*.exe", "*.vbs", "*.scr",
             "*.pif", "*.hta", "*.bat", "*.zip" ]
        {
            # these attachment types are executable
            enclose :subject "Warning" "
WARNING! The enclosed message contains executable attachments.
These attachments types may contain a computer virus program
that can infect your computer and potentently damage your data

Before clicking on these message attachments, you should verify
with the sender that this message was sent by them and not a
computer virus.
";
            break;
        }
}

```

---

### 9.3. Example 3

[TOC](#)

A Sieve script to extract subject and text out of messages from the boss

```

require ["mime", "variables", "extract_text"];

if header :contains "from" "boss@example.org"
{
  # :matches is used to get the value of the Subject header
  if header :matches "Subject" "*"
  {
    set "subject" "${1}";
  }

  # extract the first 100 bytes of the first text/* part
  for_every_part
  {
    if header :mime :type :is "Content-Type" "text"
    {
      extract_text :first 100 "msgcontent";
      break;
    }
  }

  # if it's not a 'for your information' message
  if not header :contains "subject" "FYI:"
  {
    # do something using ${subject} and ${msgcontent}
    # such as sending a notification using a notification extion
  }
}

```

---

## 10. Acknowledgements

[TOC](#)

Comments from members of the MTA Filters Working Group, in particular Ned Freed, Nigel Swinson, Mark Mallett and Alexey Melnikov, are gratefully acknowledged.

---

## 11. Security Considerations

[TOC](#)

The "enclose" action creates an entirely new message, as compared to just redirecting or forwarding the existing message. Therefore, any site policies applicable to message submission should be enforced here. The looping specification specified here provides easier access to information about the message contents, which may also be achieved

through other sieve tests. This is not believed to raise any additional security issues beyond those for the Sieve "envelope" and "body" tests. The system MUST be sized and restricted in such a manner that even malicious use of mime part matching does not deny service to other users of the host system.

Any change in a message content may interfere with digital signature mechanisms that include the body in the signed material.

All of the security considerations given in the base Sieve specification also apply to these extensions.

---

## 12. IANA Considerations

[TOC](#)

The Original-Subject: and Original-From: headers are to be registered in the Permanent Message Header Fields table.

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

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

Capability name: for\_every\_part

Description: adds the "for\_every\_part" and "break" actions for iterating through MIME parts of a message.

Capability name: mime

Description: adds ":mime" tagged arguments to the "header", "address" and "exists" commands.

Capability name: replace

Description: adds the "replace" action for replacing a MIME body part of a message.

Capability name: enclose

Description: adds the "enclose" action for enclosing a message with a wrapper.

Capability name: extract\_text

Description: adds the "extract\_text" action for extracting text from a MIME body part.

RFC number: RFC XXXX

Contact address: The Sieve discussion list <ietf-mta-filters@imc.org>. This information should be added to the list of sieve extensions given on <http://www.iana.org/assignments/sieve-extensions>.

---

[TOC](#)

## 13. Change History (to be removed prior to publication as an RFC)

---

### 13.1. draft-ietf-sieve-mime-02

[TOC](#)

minor syntax glitches in examples

Add clarification on "replace" affecting subsequent `for_every_part` loops?

Add IANA considerations for `Original-Subject:` and `Original-From:`.

Add note on "enclose" creating `From:` and `Date:` headers.

---

### 13.2. draft-ietf-sieve-mime-01

[TOC](#)

what happens when nested `for_every_loop`'s

a "mime" shorthand for testing the type/subtype, without requiring interactions with variables

notifications

notifications to calendar service

address tests, exists tests

mimeheader, mimeparameter tests

---

### 13.3. draft-ietf-sieve-mime-00

[TOC](#)

Changed title and text to emphasize MIME Tests.

Changed `for.every.part` to `for_every_part`.

Added `:anychild` to mime test. Default is to use the current context or outer envelope; specifying `:anychild` will look at all children.

Added clarifications to replacing parts affecting the structure.

Added `:mime` option to `replace`, ala `draft-ietf-sieve-vacation-06`.

Various other minor nit fixes.

---

### 13.4. draft-hansen-sieve-loop-01

[TOC](#)

Merged with `draft-daboo-sieve-mime-00.txt`.

---

### 13.5. draft-hansen-sieve-loop-02

[TOC](#)

Update to 3028bis reference.

Added 2119 conventions section.

Terminology/title tweaks.

Added informative references to body and editheader extensions.

Added description of nested loops.

Replaced mime test by extensions to header, address and exists tests.

---

### **13.6. draft-hansen-sieve-loop-03**

[TOC](#)

after enclosure, subsequent actions affect newly created message

synthesis of Date/From headers by the enclose action is no longer controversial

Filled in Security Considerations

Picked up extract\_text action from draft-ietf-sieve-notify

Expanded the IANA considerations section

---

## **14. References**

[TOC](#)

---

## 14.1. Normative References

[TOC](#)

[I-D.ietf-sieve-3028bis]	Showalter, T. and P. Guenther, " <a href="#">Sieve: An Email Filtering Language</a> ," draft-ietf-sieve-3028bis-13 (work in progress), October 2007 ( <a href="#">TXT</a> ).
[RFC2045]	Freed, N. and N. Borenstein, " <a href="#">Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies</a> ," RFC 2045, November 1996 ( <a href="#">TXT</a> ).
[RFC2047]	Moore, K., " <a href="#">MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text</a> ," RFC 2047, November 1996 ( <a href="#">TXT</a> , <a href="#">HTML</a> , <a href="#">XML</a> ).
[RFC2119]	Bradner, S., " <a href="#">Key words for use in RFCs to Indicate Requirement Levels</a> ," BCP 14, RFC 2119, March 1997 ( <a href="#">TXT</a> , <a href="#">HTML</a> , <a href="#">XML</a> ).
[RFC2822]	Resnick, P., " <a href="#">Internet Message Format</a> ," RFC 2822, April 2001 ( <a href="#">TXT</a> ).

---

## 14.2. Informative References

[TOC](#)

[I-D.ietf-sieve-body]	Guenther, P. and J. Degener, " <a href="#">Sieve Email Filtering: Body Extension</a> ," draft-ietf-sieve-body-09 (work in progress), March 2008 ( <a href="#">TXT</a> ).
[I-D.ietf-sieve-edithheader]	Guenther, P. and J. Degener, " <a href="#">Sieve Email Filtering: Edithheader Extension</a> ," draft-ietf-sieve-edithheader-11 (work in progress), March 2008 ( <a href="#">TXT</a> ).
[I-D.ietf-sieve-variables]	Homme, K., " <a href="#">Sieve Extension: Variables</a> ," draft-ietf-sieve-variables-08 (work in progress), December 2005 ( <a href="#">TXT</a> ).

---

## Authors' Addresses

[TOC](#)

	Tony Hansen
	AT&T Laboratories
	200 Laurel Ave.
	Middletown, NJ 07748
	USA
Email:	<a href="mailto:tony+sieve@loop.maillennium.att.com">tony+sieve@loop.maillennium.att.com</a>
	Cyrus Daboo
	Apple Computer, Inc.
	1 Infinite Loop
	Cupertino, CA 95014



	USA
Email:	<a href="mailto:cyrus@daboo.name">cyrus@daboo.name</a>
URI:	<a href="http://www.apple.com/">http://www.apple.com/</a>

---

## Full Copyright Statement

[TOC](#)

Copyright © The IETF Trust (2007).

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

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

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

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

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