Sieve Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 23, 2011

A. Melnikov
Isode Limited
B. Leiba
Huawei Technologies

April 21, 2011

Sieve Extension: Externally Stored Lists
draft-ietf-sieve-external-lists-07

Abstract

The Sieve scripting language can be used to implement whitelisting, blacklisting, personal distribution lists, and other sorts of list matching. Currently, this requires that all members of such lists be hardcoded in the script itself. Whenever a member of a list is added or deleted, the script needs to be updated and possibly uploaded to a mail server.

This document defines a Sieve extension for accessing externally stored lists -- lists whose members are stored externally to the script, such as using LDAP (RFC 4510), ACAP (RFC 2244), CardDAV (work in progress), or relational databases.

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 October 23, 2011.

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

This document specifies an extension to the Sieve language [RFC5228] for checking membership in an external list or for redirecting messages to an external list of recipients. An "external list" is a list whose members are stored externally to the Sieve script, such as using LDAP [RFC4510], ACAP [RFC2244], CardDAV [I-D.ietf-vcarddav-carddav], or relational databases.

This extension adds a new match type to apply to supported tests, and a new tagged argument to the "redirect" action.

1.1. Conventions Used In This Document

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

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. Extlists Extension

2.1. Capability Identifier

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

2.2. :list Match Type for Supported Tests

ABNF:

```
   MATCH-TYPE  =/ ":list"
        ; only valid for supported tests
```

The new ":list" match type changes the interpretation of the "key-list" parameter (the second parameter) in supported tests. When the match type is ":list", the key-list becomes a list of names of externally stored lists. The external lists are queried, perhaps through a list-specific mechanism, and the test evaluates to "true" if any of the specified values matches any member of one or more of the lists.

Comparators are not allowed together with the ":list" match type, so if both are specified in a test, that MUST result in an error. Queries done through list-specific mechanisms might have the effect of built-in comparators; for example, queries to certain lists might

be case-sensitive, while queries to other lists might be done without regard to case.

Implementations MUST support the use of :list in "address", "envelope" and "header" tests. Implementations that include the Variables extension [RFC5229] MUST also support its use in "string" tests.

Implementations MAY support other tests but MUST raise an error (which SHOULD be a compile-time error, but MAY be a runtime error) when a script uses :list with a test for which it is not supported. To maintain interoperability, other tests that can be used with :list SHOULD be documented in a specification that defines a capability string that can be tested (in a "require" statement, or using ihave [RFC5463]).

For example, testing 'header ["to", "cc"]' against a list would cause each "to" and "cc" value, ignoring leading and trailing whitespace, to be queried. If any value is found to belong to the list, the test returns "true". If no value belongs to the list, the test returns "false". Once a value is found in the list, there is no need for the query mechanism to look further.

For some lists, the Sieve engine might directly retrieve the list and make its own comparison. Other lists might not work that way -- they might provide a way to ask if a value is in the list, but not permit retrieval of the list itself. It is up to the Sieve implementation to understand how to interact with any supported list. If the Sieve engine is permanently unable to query the list (perhaps because the list doesn't support the required operation), the test MUST result in a runtime error in the Sieve script.

See Section 2.5 for the detailed description of syntax used for naming externally stored lists.

The :list match type uses the concept of "match variables" as defined in Section 3.2 of the Variables extension [RFC5229]. Implementations that also support that extension MUST set the ${0} match variable to the value in the list that matched the query. Other numbered match variables (${1}, ${2}, and so on) MAY be set with list-specific information that might be of use to the script.

2.3. :list Tagged Argument to the "redirect" Action

Usage: redirect :list <ext-list-name: string>

The "redirect" action with the ":list" argument is used to send the message to the set of email addresses in the externally stored list named by the ext-list-name string. This variant of the redirect command can be used to implement a personal distribution list.

For this feature to work, one of the following conditions has to be true:

1. The list resolves to a list of email addresses, and the Sieve engine is able to enumerate those addresses.

2. The list handler is able to take care of the redirection on behalf of the Sieve engine.

In cases where, for example, a list contains hashed email address values or an email address pattern ("sz*@example.com", "*+ietf@example.net"), the Sieve engine will not be able to redirect to that list, and responsibility must pass to the list handler.

If neither the Sieve engine nor the list handler can enumerate (or iterate) the list, or the list does not resolve to email addresses, the situation MUST result in a runtime error in the Sieve script.

See Section 2.5 for the detailed description of syntax used for naming externally stored lists.

2.4. Other Uses for External Lists

The uses for external lists specified here represent the useful cases and situations at the time of this writing. Other uses for external lists, using other Sieve features, might be devised in the future, and such uses can be described in extensions to this document.

2.5. Syntax of an Externally Stored List Name

A name of an externally stored list is always an absolute URI [RFC3986]. Implementations might find URIs such as LDAP [RFC4510], CardDAV [I-D.ietf-vcarddav-carddav], or Tag [RFC4151] to be useful for naming external lists.

The "tag" URI scheme [RFC4151] can be used to represent opaque, but user friendlier identifiers. Resolution of such identifiers is going to be implementation specific and it can help in hiding the complexity of an implementation from end users. For example, an implementation can provide a web interface for managing lists of users stored in LDAP. Requiring users to know generic LDAP URI

syntax might not be very practical, due to its complexity. An implementation can instead use a fixed tag URI prefix such as "tag:example.com,<date>:" (where <date> can be, for example, a date generated once on installation of the web interface and left untouched upon upgrades) and the prefix doesn't even need to be shown to end users.

The "ab" URI scheme (in particular, the URI "ab:default"), defined in Section 4.3 MUST be supported. The mandatory-to-implement URI "ab:default" gives access to the user's default address book (usually the user's personal address book).

It's possible that a server will have no access to anything resembling an address book (perhaps in an implementation where address books are only client-side things), but the server can still provide access to other sorts of lists -- consider the list of dates in Example 2 (Section 2.8.2), or lists of important keywords and the like. It might sometimes make sense to map "ab:default" into some available list, but that might not always be reasonable. If there really is no concept of an address book in a particular server implementation, the server MAY support "ab:default" by having all matches to it fail. Such an implementation SHOULD NOT be done except as a last resort.

Queries against address books SHOULD be done without regard to case.

2.6. Test valid_ext_list

Usage: valid_ext_list <ext-list-names: string-list>

The "valid_ext_list" test is true if all of the external list names in the ext-list-names argument are supported, and they are valid both syntactically (including URI parameters) and semantically (including implementation-specific semantic restrictions). Otherwise the test returns false.

This test MUST perform exactly the same validation of an external list name as would be performed by the "header :list" test.

2.7. Interaction with ManageSieve

This extension defines the following new capability for ManageSieve (see [RFC5804] section 1.7):

EXTLISTS - A space-separated list of URI schema parts [RFC3986] for supported externally stored list types. This capability MUST be returned if the corresponding Sieve implementation supports the "extlists" extension defined in this document.

This also extends the ManageSieve ABNF as follows:

```
single-capability  =/ DQUOTE "EXTLISTS" DQUOTE SP ext-list-types CRLF
        ; single-capability is defined in [RFC5804]

ext-list-types  = string
        ; space separated list of URI schema parts
        ; for supported externally stored list types.
        ; MUST NOT be empty.
```

2.8. Examples

2.8.1. Example 1

This example uses a personal address book, along with the Spamtest [RFC5235] and Relational [RFC5231] extensions to give a different level of spam tolerance to known senders.

```
require ["envelope", "extlists", "fileinto", "spamtest",
         "relational", "comparator-i;ascii-numeric"];
if envelope :list "from" "ab:default"
  { /* Known: allow high spam score */
    if spamtest :value "ge" :comparator "i;ascii-numeric" "8"
      {
        fileinto "spam";
      }
  }
elsif spamtest :value "ge" :comparator "i;ascii-numeric" "3"
  { /* Unknown: less tolerance in spam score */
    fileinto "spam";
  }
```

The same example can also be written another way, if the Variables extension [RFC5229] is also supported:

```
require ["envelope", "extlists", "fileinto", "spamtest",
    "variables", "relational", "comparator-i;ascii-numeric"];
if envelope :list "from" "ab:default" {
  set "lim" "8";  /* Known: allow high spam score */
} else {
  set "lim" "3";  /* Unknown: less tolerance in spam score */
}
if spamtest :value "ge" :comparator "i;ascii-numeric" "${lim}" {
  fileinto "spam";
}
```

2.8.2. Example 2

This example uses the "currentdate" test [RFC5260] and a list containing the dates of local holidays. If today is a holiday, the script will notify [RFC5435] the user via XMPP [RFC5437] about the message.

```
require ["extlists", "date", "enotify"];
if currentdate :list "date"
   "tag:example.com,2011-01-01:localHolidays" {
   notify "xmpp:romeo@im.example.com";
}
```

2.8.3. Example 3

This example also uses the "envelope" option [RFC5228] and the Subaddress extension [RFC5233]. If mail is sent with the list name as a subaddress of the recipient (to, say, "alexey+mylist"), and the message comes from a member of the list, it will be redirected to all members of the list. Variants of this technique might be useful for creating private mailing lists.

```
require ["extlists", "envelope", "subaddress"];

# Submission from list members is sent to all members
if allof (envelope :detail "to" "mylist",
          header :list "from"
                 "tag:example.com,2010-05-28:mylist") {
    redirect :list "tag:example.com,2010-05-28:mylist";
}
```

2.8.4. Example 4

This example uses variable matching [RFC5229] to extract the IP address from the last "Received" header field. It then checks that against a "block list" of undesirable IP addresses, and rejects the message if there's a match.

```
require ["variables", "extlists", "index", "reject"];
if header :index 1 :matches "received" "*(* [*.*.*.*])*" {
  set "ip" "${3}.${4}.${5}.${6}";
  if string :list "${ip}"
      "tag:example.com,2011-04-10:DisallowedIPs" {
    reject "Message not allowed from this IP address";
  }
}
```

2.8.5. Example 5

This example uses several features of the MIME parts extension [RFC5703] to scan for unsafe attachment types. To make it easily extensible, the unsafe types are kept in an external list, which would be shared among all users and all scripts, avoiding the need to change scripts when the list changes.

[Note that this is an illustrative example, and more rigorous malware filtering is advisable. It is insufficient to base email security on checks of filenames alone.]

```
    require [ "extlists", "foreverypart", "mime", "enclose" ];

    foreverypart
    {
      if header :mime :param "filename"
         :list ["Content-Type", "Content-Disposition"]
           "tag:example.com,2011-04-10:BadFileNameExts"
      {
        # these attachment types are executable
        enclose :subject "Warning" :text
 WARNING! The enclosed message attachments that might be unsafe.
 These attachment types may contain a computer virus program
 that can infect your computer and potentially damage your data.

 Before clicking on these message attachments, you should verify
 with the sender that this message was sent intentionally, and
 that the attachments are safe to open.
 .
 ;
        break;
      }
    }
```

3. Security Considerations

Security considerations related to the "address"/"envelope"/"header" tests and "redirect" action discussed in Sieve [RFC5228] also apply to this document.

External list memberships ought to be treated as if they are an integral part of the script, so a temporary failure to access an external list SHOULD be handled in the same way as a temporary failure to retrieve the Sieve script itself.

For example, if the Sieve script is stored in the Lightweight

Directory Access Protocol [RFC4510] and the script can't be retrieved when a message is processed (perhaps the LDAP server is unavailable), then the Sieve engine might delay message delivery until the script can be retrieved successfully. Similarly, if an external list is stored in LDAP and that LDAP server is unavailable, the Sieve engine would take the same action -- delay message delivery and try again later.

Protocols/APIs used to retrieve/verify external list membership MUST provide an appropriate level of confidentiality and authentication. Usually, that will be at least the same level of confidentiality as protocols/APIs used to retrieve Sieve scripts, but only the implementation (or deployment) will know what is appropriate. There's a difference, for example, between making an LDAP request on a closed LAN that's only used for trusted servers (it may be that neither encryption nor authentication is needed), on a firewalled LAN internal to a company (it might be OK to skip encryption, depending upon policy), and on the open Internet (encryption and authentication are probably both required). It also matters whether the list being accessed is private or public (no encryption or authentication may be needed for public data, even on the Internet).

Implementations of this extension should keep in mind that matching values against an externally stored list can be IO and/or CPU intensive. This can be used to deny service to the mailserver and/or to servers providing access to externally stored mailing lists. A naive implementation, such as the one that tries to retrieve content of the whole list to perform matching can make this worse.

But note that many protocols that can be used for accessing externally stored lists support flexible searching features that can be used to minimize network traffic and load on the directory service. For example, LDAP allows for search filters. Implementations SHOULD use such features whenever they can.

Many organizations support external lists with thousands of recipients. In order to avoid mailbombs when redirecting a message to an externally stored list, implementations SHOULD enforce limits on the number of recipients and/or on domains to which such recipients belong.

Note in particular that it can be too easy for a script to use

```
redirect :list "ab:default";
```

to send messages to "everyone in your address book", and one can easily imagine both intentional and accidental abuse. The situation can be even worse for, say, "ab:corporate". Warnings, as well as enforced limits, are appropriate here.

4. IANA Considerations

4.1. Registration of Sieve Extension

The following template specifies the IANA registration of the Sieve extension specified in this document. This information should be added to the list of sieve extensions given on http://www.iana.org/assignments/sieve-extensions.

To: iana@iana.org

Subject: Registration of new Sieve extension

Capability name: extlists

Description: Adds the ":list" match type to certain Sieve tests, and the ":list" argument to the "redirect" action. The ":list" match type changes tests to match values against values stored in one or more externally stored lists. The ":list" argument to the redirect action changes the redirect action to forward the message to email addresses stored in the externally stored list.

RFC number: this RFC

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

4.2. Registration of ManageSieve Capability

The following requests IANA to register a new ManageSieve Capability according to the IANA registration template specified in [RFC5804]:

To: iana@iana.org

Subject: ManageSieve Capability Registration

Capability name: extlists

Description: This capability is returned if the server supports the "extlists" [RFCXXXX] Sieve extension.

Relevant publications: this RFC, Section 2.7

Person & email address to contact for further information: Sieve mailing list <sieve@ietf.org>

Author/Change controller: IESG

4.3. Registration of "ab" URI Scheme

The following requests IANA to register a new URI scheme according to the IANA registration template specified in [RFC4395]:

URI scheme name: ab

Status: Permanent

URI scheme syntax:

```
paburi = "ab:" addrbook [ "?" extensions ]
addrbook = segment
     ; <segment> defined in [RFC3986]
extensions = query
     ; <query> defined in [RFC3986]
```

URI scheme semantics: "ab" URIs are used for designating references to address books. An address book is an internal concept used by different applications (such as Sieve interpreters) for describing a list of named entries, and may be translated into other types of address books, such as LDAP Groups. Address books may be private or shared; they may be personal, organizational, or perhaps even "crowdsourced".

Encoding considerations: Percent-encoding is allowed in "segment" and "query" components. Internationalization is handled by IRI processing.

Intended usage: An "ab" URI is designed to be used internally by applications for referencing address books. Each URI is intended to represent a grouping of addresses that can be logically thought of as one "book". Any given address can belong to more than one book -- that is, can be referred to by more than one URI.

The URI "ab:default" is a reserved name that MUST be implemented, representing a default grouping (book) of addresses. Other names, representing the same or other groupings MAY be implemented. For example, an implementation might have the following URIs:

* ab:personal -- a book representing the user's personal address book.

* ab:friends -- a subset of ab:personal, defined by the user.

* ab:family -- a subset of ab:personal, defined by the user.

* ab:company -- a book representing user's company's address book.

* ab:department -- a subset of ab:company, defined by the company.

* ab:co-workers -- a subset of ab:company, defined by the user.

* ab:default -- the default address book, a reference to ab:personal.

Applications and/or protocols that use this URI scheme name:
Currently only the Sieve External List extension is using this URI scheme. Email clients that use URIs internally might find this URI scheme to be useful as well.

Interoperability considerations: Applications are only REQUIRED to support "ab:default".

Security considerations: Applications SHOULD ensure appropriate restrictions are in place to protect sensitive information that might be revealed by "ab" URIs from access or modification by untrusted sources.

Relevant publications: this RFC

Contact: Sieve mailing list <sieve@ietf.org>

Author/Change controller: IETF/IESG

5. Acknowledgements

Thanks to Alexandros Vellis, Nigel Swinson, Ned Freed, Kjetil Torgrim Homme, Dave Cridland, Cyrus Daboo, Pete Resnick, and Robert Burrell Donkin for ideas, comments and suggestions. Kristin Hubner also helped greatly with the examples.

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.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4151] Kindberg, T. and S. Hawke, "The 'tag' URI Scheme", RFC 4151, October 2005.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.

[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.

[RFC5804] Melnikov, A. and T. Martin, "A Protocol for Remotely Managing Sieve Scripts", RFC 5804, July 2010.

6.2. Informative References

[I-D.ietf-vcarddav-carddav] Daboo, C., "vCard Extensions to WebDAV (CardDAV)", draft-ietf-vcarddav-carddav-10 (work in progress), November 2009.

[RFC2244] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

[RFC4510] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

[RFC5229] Homme, K., "Sieve Email Filtering: Variables Extension", RFC 5229, January 2008.

[RFC5231] Segmuller, W. and B. Leiba, "Sieve Email Filtering: Relational Extension", RFC 5231, January 2008.

[RFC5233] Murchison, K., "Sieve Email Filtering: Subaddress Extension", RFC 5233, January 2008.

[RFC5235] Daboo, C., "Sieve Email Filtering: Spamtest and Virustest Extensions", RFC 5235, January 2008.

[RFC5260] Freed, N., "Sieve Email Filtering: Date and Index Extensions", RFC 5260, July 2008.

[RFC5435] Melnikov, A., Leiba, B., Segmuller, W., and T. Martin, "Sieve Email Filtering: Extension for Notifications", RFC 5435, January 2009.

[RFC5437] Saint-Andre, P. and A. Melnikov, "Sieve Notification Mechanism: Extensible Messaging and Presence Protocol (XMPP)", RFC 5437, January 2009.

[RFC5463] Freed, N., "Sieve Email Filtering: Ihave Extension", RFC 5463, March 2009.

[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

Barry Leiba
Huawei Technologies

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