Internet Engineering Task Force (IETF)
Request for Comments: 6134
Category: Standards Track
ISSN: 2070-1721

A. Melnikov
Isode Limited
B. Leiba
Huawei Technologies

July 2011

Sieve Extension: Externally Stored Lists

Abstract

The Sieve email filtering language can be used to implement email whitelisting, blacklisting, personal distribution lists, and other sorts of list matching. Currently, this requires that all members of such lists be hard-coded 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 the Lightweight Directory Access Protocol (LDAP), the Application Configuration Access Protocol (ACAP), vCard Extensions to WebDAV (CardDAV), or relational databases.

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/rfc6134.

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 [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 ABNF [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 than the ones in this document. Implementations MUST report an error when a script uses ":list" with a test that does not support ":list". This error SHOULD be reported at compile-time but MAY be reported at run-time. 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 known cases and situations at the time of this writing. Other uses for external lists, employing 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 [CardDAV], or "tag" [RFC4151] to be useful for naming external lists.

The "tag" URI scheme [RFC4151] can be used to represent opaque, but more user-friendly 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 "addrbook" URNs defined in Section 2.6 (in particular, the reserved URI "urn:ietf:params:sieve:addrbook:default") MUST be supported. To make it easier to use registered Sieve URN parameters, we define a shorthand way to specify them in a Sieve script: a list name that begins with ":" is taken as referencing a Sieve URN parameter, with the initial ":" expanding to "urn:ietf:params:sieve:". So we have the following equivalences:

```
:addrbook:default == urn:ietf:params:sieve:addrbook:default

:addrbook:personal == urn:ietf:params:sieve:addrbook:personal
```

and so on.

The mandatory-to-implement URI

```
urn:ietf:params:sieve:addrbook:default
```

gives access to the user's default address book (usually the user's personal address book). Note that these are URIs, subject to normal URI encoding rules, including percent-encoding. The reserved name "default" MUST be considered case-insensitive after decoding. That means that the following URIs are all equivalent:

```
:addrbook:default

:ADDRBOOK:DEFAULT

:aDdRbOOk:DeFauLt

:AddrBook:%44%65%66ault
```

Address book names other than "default" MAY be case-sensitive, depending upon the implementation, so their case (after URI decoding) MUST be maintained.

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 2.9.2), or lists of important keywords and the like. It might sometimes make sense to map ":addrbook: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 ":addrbook: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. Definition of "addrbook" URN Parameter

This section gives the details of the "addrbook" Sieve URN parameter that's registered in Section 4.4. URIs that use this parameter begin with "urn:ietf:params:sieve:addrbook:".

URN parameter name: addrbook

URN parameter syntax: The "addrbook" parameter is defined by the <addrbook-urn> rule, defined using ABNF [RFC5234]:

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

Intended usage: "addrbook" URNs are used for designating references to address books. An address book is a 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".

The address book name (the "addrbook" element in the ABNF above) refers to a specifically named address book, as defined by the implementation. A user might, for example, have access to a number of different address books, such as a personal one, a family one, a company one, and one for the town where the user lives.

The extension information (the "extensions" element in the ABNF above) is available for use in future extensions. It might allow for things such as dynamic subsets of an address book -- for example, something such as this might be defined in the future:

urn:ietf:params:sieve:addrbook:personal?name.contains=fred

There are no extensions defined at this time.

An "addrbook" URN is designed to be used by applications for referencing address books. Each URN 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 URN.

The URI "urn:ietf:params:sieve:addrbook" has no meaning in itself. It MUST be used with sub-parameters representing the address book name and extension information, as shown in the ABNF above.

The sub-parameter "default" (creating the URN "urn:ietf:params:sieve:addrbook:default") is a reserved (case-insensitive) 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 use the following sub-parameters:

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

* friends -- a subset of urn:ietf:params:sieve:addrbook:personal, defined by the user.

* family -- a subset of urn:ietf:params:sieve:addrbook:personal, defined by the user.

* company -- a book representing the user's company's address book.

* department -- a subset of urn:ietf:params:sieve:addrbook:company, defined by the company.

* co-workers -- a subset of urn:ietf:params:sieve:addrbook:company, defined by the user.

* default -- the default address book, a reference to urn:ietf:params:sieve:addrbook:personal.

Interoperability considerations: Applications are only REQUIRED to support "addrbook:default", where all cases and encodings of "default" are considered equivalent. Address book names other than "default" MAY be case-sensitive, depending upon the

implementation, so their case (after URI decoding) MUST be maintained.

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

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

2.7. 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.

2.8. 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.
```

2.9. Examples

2.9.1. 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" ":addrbook: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" ":addrbook: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";
}
```

2.9.2. 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 (as described in [RFC5435]) the user via the Extensible Messaging and Presence Protocol (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";
}
```

2.9.3. 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";
}
```

2.9.4. 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";
  }
}
```

2.9.5. 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 has 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).

Having the processing and outcome of a Sieve script depend on the contents of external data can allow someone with control of the external data to have unusual, and perhaps unauthorized, control of the script -- and, consequently, of the disposition of the user's email. A user using such a list for spam control, for example, might find important mail being discarded because of tampering with the list. Someone using redirect to an external list could have her email redirected to the wrong eyes because of such tampering. Security and integrity protection of external lists is as important as protection of the Sieve script itself.

Implementations of this extension should keep in mind that matching values against an externally stored list can be I/O and/or CPU intensive. This can be used to deny service to the mail server 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 ":addrbook: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, ":addrbook:corporate". Warnings, as well as enforced limits, are appropriate here.

Applications SHOULD ensure appropriate restrictions are in place to protect sensitive information that might be revealed by "addrbook" URNs from access or modification by untrusted sources.

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 has been added to the Sieve Extensions registry on http://www.iana.org.

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: RFC 6134

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

4.2. Registration of ManageSieve Capability

IANA has registered 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" RFC 6134 Sieve extension.

Relevant publications: RFC 6134, Section 2.8

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

Author/Change controller: IESG

4.3. Creation of Sieve URN Parameters Registry

IANA has created a new registry under "Sieve Extensions" for Sieve URN Parameters. Registration into this registry is according to the "Specification Required" policy [RFC5226].

The registry contains the following two items:

URN parameter name: The name of the URN parameter. If the name is "paramname", the resulting top-level URN will be "urn:ietf:params:sieve:paramname".

Reference: The document and section where the definition of the parameter can be found. Be sure to include the section number as well as the document reference, so the documentation is easy to find.

The documentation -- which will be in the referenced document and section, and will not be included in the registry -- MUST include the following information (see Section 2.6 for an example):

URN parameter name: The name of the URN parameter.

URN parameter syntax: The syntax of the parameter and any sub-parameters, which SHOULD be specified using ABNF [RFC5234].

Intended usage: A detailed description of how the parameter and any sub-parameters are expected to be used. This is the place to define static sub-parameters, registries for sub-parameters, options, registries for options, and so on.

Interoperability considerations: Any notes specific to interoperability issues. This is where to put mandatory-to-implement sub-parameters and the like.

Security considerations: Any notes specific to security and privacy issues.

Contact: Contact information, in case there are questions.

4.4. Registration of the "addrbook" URN parameter

IANA has registered a new Sieve URN parameter in the registry defined in Section 4.3.

URN parameter name: addrbook

Reference: RFC 6134, Section 2.6

4.5. Registration of "sieve" URN sub-namespace

IANA has registered a new URN sub-namespace within the IETF URN Sub-namespace for Registered Protocol Parameter Identifiers defined in [RFC3553].

Registry name: sieve

Specification: RFC 6134

Repository: Sieve URN Parameters registry (Section 4.3)

Index value: Sub-parameters MUST be specified in UTF-8, using standard URI encoding where necessary.

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.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[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

[CardDAV] Daboo, C., "vCard Extensions to WebDAV (CardDAV)", Work in Progress, November 2009.

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

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, June 2003.

[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/