Network Working Group
Request for Comments: 5435
Category: Standards Track

A. Melnikov, Ed.
Isode Limited
B. Leiba, Ed.
W. Segmuller
IBM T.J. Watson Research Center
T. Martin
Endless Crossword

January 2009

Sieve Email Filtering: Extension for Notifications

Status of This Memo

Copyright Notice

Abstract

Users go to great lengths to be notified as quickly as possible that they have received new mail. Most of these methods involve polling to check for new messages periodically. A push method handled by the final delivery agent gives users quicker notifications and saves server resources. This document does not specify the notification method, but it is expected that using existing instant messaging infrastructure such as Extensible Messaging and Presence Protocol (XMPP), or Global System for Mobile Communications (GSM) Short Message Service (SMS) messages will be popular. This document describes an extension to the Sieve mail filtering language that allows users to give specific rules for how and when notifications should be sent.

Table of Contents

1. Introduction

This is an extension to the Sieve language defined by [Sieve] for providing instant notifications. It defines the new action "notify".

This document does not specify the notification methods. Examples of possible notification methods are email and XMPP. To allow for the portability of scripts that use notifications, implementation of the [MailTo] method is mandatory. Other available methods shall depend upon the implementation and configuration of the system.

1.1. Conventions Used in This Document

Conventions for notations are as in [Sieve], Section 1.1, including the use of [ABNF].

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 [Kwds].

2. Capability Identifier

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

3. Notify Action

3.1. Notify Action Syntax and Semantics

```
Usage:  notify [":from" string]
        [":importance" <"1" / "2" / "3">]
        [":options" string-list]
        [":message" string]
        <method: string>
```

The "notify" action specifies that a notification should be sent to a user. The format of the notification is implementation-defined and is also affected by the notification method used (see Section 3.2). However, all content specified in the ":message" parameter SHOULD be included.

3.2. Notify Parameter "method"

The "method" positional parameter identifies the notification method that will be used; it is a URI [URI]. For example, the notification method can be a tel URI [TEL-URI] with a phone number to send SMS messages to, or an XMPP [XMPP] URI containing an XMPP identifier [XMPP-URI].

The supported URI values will be site-specific, but support for the [MailTo] method is REQUIRED in order to ensure interoperability. If a URI schema is specified that the implementation does not support, the notification MUST cause an error condition at run time. Sieve scripts can check the supported methods using the valid_notify_method test to be sure that they only use supported ones, to avoid such error conditions.

If the "method" parameter contains a supported URI schema, then the URI MUST be checked for syntactic validity. Invalid URI syntax or an unsupported URI extension MUST cause an error. An implementation MAY enforce other semantic restrictions on URIs -- for example, to restrict phone numbers in a tel: URI to a particular geographical region -- and will treat violations of such semantic restrictions as errors.

3.3. Notify Tag ":from"

A ":from" tag may be used to specify an author of the notification. The syntax of this parameter's value is method-specific. Implementations SHOULD check the syntax according to the notification method specification and generate an error when a syntactically invalid ":from" tag is specified.

In order to minimize/prevent forgery of the author value, implementations SHOULD impose restrictions on what values can be specified in a ":from" tag. For example, an implementation may restrict this value to be a member of a list of known author addresses or to belong to a particular domain. It is suggested that values that don't satisfy such restrictions simply be ignored rather than causing the "notify" action to fail.

3.4. Notify Tag ":importance"

The ":importance" tag specifies the importance of quick delivery of the notification, as perceived by the Sieve script owner. The ":importance" tag is followed by a numeric value represented as a string: "1" (high importance), "2" (normal importance), and "3" (low importance). If no importance is given, the default value "2" SHOULD be assumed. A notification method MAY treat the importance value as a transport indicator. For example, it might deliver notifications of high importance quicker than notifications of normal or low importance. Some notification methods allow users to specify their state of activity (for example, "busy" or "away from keyboard"). If the notification method provides this information, it SHOULD be used to selectively send notifications. If, for example, the user marks

herself as "busy", a notification method can require that a notification with importance of "3" is not to be sent; however, the user might be notified of a notification with higher importance.

If the notification method allows users to filter messages based upon certain parameters in the message, users SHOULD be able to filter based upon importance. If the notification method does not support importance, then this parameter MUST be ignored. An implementation MAY include the importance value in the default message, Section 3.6, if one is not provided.

3.5. Notify Tag ":options"

The ":options" tag is used to send additional parameters to the notification method. Interpretation of the parameters is method-specific. This document doesn't specify any such additional parameter.

Each string in the options string list has the following syntax: "<optionname>=<value>"
where optionname has the following ABNF [ABNF]:

```
l-d = ALPHA / DIGIT
l-d-p = l-d / "." / "-" / "_"
optionname = l-d *l-d-p
value = *(%x01-09 / %x0B-0C / %x0E-FF)
```

3.6. Notify Tag ":message"

The ":message" tag specifies the message data to be included in the notification. The entirety of the string SHOULD be sent, but implementations MAY shorten the message for technical or aesthetic reasons. If the ":message" parameter is absent, a default implementation-specific message is used. Unless otherwise specified by a particular notification mechanism, an implementation default containing at least the value of the "From" header field and the value of the "Subject" header field is RECOMMENDED.

In order to construct more complex messages, the notify extension can be used together with the Sieve variables extension [Variables], as shown in the examples below.

3.7. Examples

Example 1:

```
require ["enotify", "fileinto", "variables"];

if header :contains "from" "boss@example.org" {
    notify :importance "1"
        :message "This is probably very important"
                    "mailto:alm@example.com";
    # Don't send any further notifications
    stop;
}

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

    # :matches is used to get the value of the From header
    if header :matches "From" "*" {
        set "from" "${1}";
    }

    notify :importance "3"
        :message "[SIEVE] ${from}: ${subject}"
        "mailto:alm@example.com";
    fileinto "INBOX.sieve";
}
```

Example 2:

```
require ["enotify", "fileinto", "variables", "envelope"];

if header :matches "from" "*@*.example.org" {
    # :matches is used to get the MAIL FROM address
    if envelope :all :matches "from" "*" {
        set "env_from" " [really: ${1}]";
    }

    # :matches is used to get the value of the Subject header
    if header :matches "Subject" "*" {
        set "subject" "${1}";
    }

    # :matches is used to get the address from the From header
    if address :matches :all "from" "*" {
        set "from_addr" "${1}";
    }
```

```
           notify :message "${from_addr}${env_from}: ${subject}"
                           "mailto:alm@example.com";
       }
```

Example 3:

```
    require ["enotify", "variables"];

    set "notif_method"
    "xmpp:tim@example.com?message;subject=SIEVE;body=You%20got%20mail";

    if header :contains "subject" "Your dog" {
        set "notif_method" "tel:+14085551212";
    }

    if header :contains "to" "sievemailinglist@example.org" {
        set "notif_method" "";
    }

    if not string :is "${notif_method}" "" {
        notify "${notif_method}";
    }

    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}";
        }

        # don't need high importance notification for
        # a 'for your information'
        if not header :contains "subject" "FYI:" {
            notify :importance "1" :message "BOSS: ${subject}"
                                "tel:+14085551212";
        }
    }
```

3.8. Requirements on Notification Methods Specifications

This section describes requirements for documents that define specific Sieve notification methods.

Notification mechanisms MUST NOT add new Sieve tags to the "notify" action.

A notification method MAY allow modification of the final notification text -- for example, truncating it if it exceeds a length limit or modifying characters that can not be represented in the target character set. Characters in the notification text that

can't be represented by the notification method SHOULD be replaced with a symbol indicating an unknown character. Allowed modifications MUST be documented in the document describing the notification method.

A notification method MAY ignore parameters specified in the "notify" action.

A notification method MAY recommend the default message value to be used if the ":message" argument is not specified.

Notifications SHOULD include timestamps, if the notification method allows for their transmission outside of the textual message. Implementation methods that can only transmit timestamps in the textual message MAY include them in the textual message.

A notification MUST include means to identify/track its origin in order to allow a recipient to stop notifications or find out how to contact the sender. This requirement is to help with tracking a misconfigured or abusive origin of notifications.

Methods SHOULD NOT include any other extraneous information not specified in parameters to the "notify" action.

Methods MUST specify which URI parameters (if any) must be ignored, which ones must be used in the resulting notification, and which ones must cause an error.

Methods MUST specify what values are returned by the notify_method_capability test, Section 5, in particular for the "online" notification-capability.

If there are errors sending the notification, the Sieve interpreter SHOULD ignore the notification and not retry indefinitely. The Sieve interpreter MAY throttle notifications; if it does, a request to send a notification MAY be silently ignored. Documents describing notification methods SHOULD describe how retries, throttling, duplicate suppression (if any), etc. are to be handled by implementations.

4. Test valid_notify_method

Usage: valid_notify_method <notification-uris: string-list>

The valid_notify_method test is true if the notification methods listed in the notification-uris argument are supported and they are valid both syntactically (including URI parameters) and semantically

(including implementation-specific semantic restrictions). This test MUST perform exactly the same validation as would be performed on the "method" parameter to the "notify" action.

The test is true only if ALL of the listed notification methods are supported and valid.

Example 4 (partial):

```
if not valid_notify_method ["mailto:",
        "http://gw.example.net/notify?test"] {
    stop;
}
```

5. Test notify_method_capability

Usage: notify_method_capability [COMPARATOR] [MATCH-TYPE]
<notification-uri: string>
<notification-capability: string>
<key-list: string-list>

The notify_method_capability test retrieves the notification capability specified by the notification-capability string that is specific to the notification-uri and matches it to the values specified in the key-list. The test succeeds if a match occurs. The type of match defaults to ":is", and the default comparator is "i;ascii-casemap".

The notification-capability parameter is case insensitive.

The notify_method_capability test MUST fail unconditionally if the specified notification-uri is syntactically invalid (as determined by the valid_notify_method test, Section 4) or specifies an unsupported notification method. However this MUST NOT cause an error.

The notify_method_capability test MUST fail unconditionally if the specified notification-capability item is not known to the Sieve interpreter. A script MUST NOT fail with an error if the item does not exist. This allows scripts to be written that handle nonexistent items gracefully.

This document defines a single notification-capability value "online", which is described below. Additional notification-capability values may be defined by using the procedure defined in Section 9.3.

The "relational" extension [Relational] adds a match type called ":count". The count of an notify_method_capability test is 0, if the returned information is the empty string, or 1.

For the "online" notification-capability, the notify_method_capability test can match one of the following key-list values:

- "yes" - the entity identified by the notification-uri can receive a notify notification immediately. Note that even after this value is returned, there is no guarantee that the entity would actually be able to receive any notification immediately or even receive it at all. Transport errors, recipient policy, etc. can prevent that.

- "no" - the entity identified by the notification-uri is not currently available to receive an immediate notification.

- "maybe" - the Sieve interpreter can't determine if the entity identified by the notification-uri is online or not.

Example 5:

```
require ["enotify"];

if notify_method_capability
       "xmpp:tim@example.com?message;subject=SIEVE"
       "Online"
       "yes" {
    notify :importance "1" :message "You got mail"
         "xmpp:tim@example.com?message;subject=SIEVE";
} else {
    notify :message "You got mail" "tel:+14085551212";
}
```

6. Modifier encodeurl to the 'set' Action

Usage: ":encodeurl"

When the Sieve script specifies both "variables" [Variables] and "enotify" capabilities in the "require", a new "set" action modifier (see [Variables]) ":encodeurl" becomes available to Sieve scripts. This modifier performs percent-encoding of any octet in the string that doesn't belong to the "unreserved" set (see [URI]). The percent-encoding procedure is described in [URI].

The ":encodeurl" modifier has precedence 15.

Example 6:

```
require ["enotify", "variables"];

set :encodeurl "body_param" "Safe body&evil=evilbody";

notify "mailto:tim@example.com?body=${body_param}";
```

7. Interactions with Other Sieve Actions

The "notify" action is compatible with all other actions, and does not affect the operation of other actions. In particular, the "notify" action MUST NOT cancel the implicit keep.

Multiple executed "notify" actions are allowed. Specific notification methods MAY allow multiple notifications from the same script to be collapsed into one.

8. Security Considerations

Security considerations are discussed in [Sieve]. Additionally, implementations must be careful to follow the security considerations of the specific notification methods.

The "notify" action is potentially very dangerous. The path the notification takes through the network may not be secure. An error in the options string may cause the message to be transmitted to someone it was not intended for, or may expose information to eavesdroppers.

Just because a notification is received doesn’t mean that it was sent by the Sieve implementation. It might be possible to forge notifications or modify parts of valid notifications with some notification methods.

Forgery of the ":importance" value (for example, by unauthorized script modification) can potentially result in slowdown in notification delivery.

Note that some components of notifications should not be trusted. For example, the timestamp field can be easily forged or modified when some notification transports are used. Even if the timestamp is believed to be correct by the sender and is not modified in transit, it might be misleading on the receiving system due to clock differences.

An organization may have a policy about the forwarding of classified information to unclassified networks. Unless the policy is also enforced in the module responsible for the generating (or sending) of notifications, users can use the extension defined in this document to extract classified information and bypass the policy.

Notifications can result in loops and bounces. Also, allowing a single script to notify multiple destinations can be used as a means of amplifying the number of messages in an attack. Moreover, if loop detection is not properly implemented, it may be possible to set up exponentially growing notification loops. Accordingly, Sieve notification methods:

1. MUST provide mechanisms for avoiding notification loops.

2. MUST provide the means for administrators to limit the ability of users to abuse notify. In particular, it MUST be possible to limit the number of "notify" actions a script can perform. Additionally, if no use cases exist for using "notify" with multiple destinations, this limit SHOULD be set to 1. Additional limits, such as the ability to restrict "notify" to local users, MAY also be implemented.

3. MUST provide facilities to log the use of "notify" in order to facilitate tracking down abuse.

4. MAY use script analysis to determine whether or not a given script can be executed safely. While the Sieve language is sufficiently complex so that full analysis of all possible scripts is computationally infeasible, the majority of real-world scripts are amenable to analysis. For example, an implementation might allow scripts that it has determined to be safe to run unhindered, block scripts that are potentially problematic, and subject unclassifiable scripts to additional auditing and logging.

Allowing "notify" action at all may not be appropriate in situations where Sieve scripts are associated with email accounts that are freely-available and/or not trackable to a human who can be held accountable for creating message bombs or other abuse.

Implementations that construct URIs internally from various notify parameters MUST make sure that all components of such URIs are properly percent-encoded (see [URI]). In particular, this applies to values of the ":from" and ":message" tagged arguments and may apply to the ":options" values.

Header/envelope tests [Sieve], together with Sieve variables, can be used to extract the list of users to receive notifications from the incoming email message or its envelope. This is potentially quite dangerous, as this can be used for denial-of-service attacks on recipients controlled by the message sender. For this reason, implementations SHOULD NOT allow the use of variables containing values extracted from the email message in the "method" parameter to the "notify" action. Note that violation of this SHOULD NOT may result in the creation of an open relay, i.e., any sender would be able to create specially crafted email messages that would result in notifications delivered to recipients under the control of the sender. In the worst case, this might result in financial loss by the user controlling the Sieve script and/or by recipients of notifications (e.g., if a notification is an SMS message).

Note that the last SHOULD NOT is not a generic prohibition of use of variables in the "notify" action, as controlling the target of a notification by extracting it from user-owned data stores (such as user's Lightweight Directory Access Protocol (LDAP) entry) is considered to be useful.

It is imperative that whatever implementations use to store the user-defined filtering scripts protect them from unauthorized modification, to preserve the integrity of the mail system. An attacker who can modify a script can cause mail to be discarded, rejected, or forwarded to an unauthorized recipient. In addition, it's possible that Sieve scripts might expose private information, such as mailbox names or email addresses of favored (or disfavored) correspondents. Because of that, scripts SHOULD also be protected from unauthorized retrieval.

9. IANA Considerations

9.1. Registration of Sieve Extension

To: iana@iana.org
Subject: Registration of new Sieve extension
Capability name: enotify
Description: adds the "notify" action for notifying user about the
received message. It also provides two new tests:
valid_notify_method checks notification URIs for validity;
notify_method_capability can check recipients capabilities.
RFC number: this RFC
Contact address: The Sieve discussion list
<ietf-mta-filters@imc.org>

This information has been added to the list of Sieve extensions available from http://www.iana.org/.

9.2. New Registry for Sieve Notification Mechanisms

IANA has created a new registry for Sieve notification mechanisms. This registry contains both vendor-controlled notification mechanism names (beginning with "vnd.") and IETF-controlled notification mechanism names. Vendor-controlled notification mechanism names have the format as defined in the following paragraph and may be registered on a "First Come First Served" basis [IANA-GUIDELINES], by applying to IANA with the form specified later in this section. Registration of notification mechanisms that do not begin with "vnd." are registered using a "Specification Required" policy [IANA-GUIDELINES].

Vendor-controlled notification mechanism names MUST have the form "vnd.<vendor-name>.<mechanism-name>", where <vendor-name> is as specified in the Application Configuration Access Protocol (ACAP) Vendor Subtree registry [ACAP].

This defines the template for a new registry for Sieve notification mechanisms, which has been created and is available from http://www.iana.org/. There are no initial entries for this registry.

To: iana@iana.org
Subject: Registration of new Sieve notification mechanism
Mechanism name: [the name of the mechanism]
Mechanism URI: [the RFC number of the document that defines the URI used by this mechanism. Different mechanisms MUST use different URI schema.]
Mechanism-specific options: [the names of any Sieve notify options (as used in the ":options" parameter) that are specific to this mechanism, or "none"]
Permanent and readily available reference: [the RFC number or an URL of the document that defines this notification mechanism]
Person and email address to contact for further information: [the name and email address of the technical contact for information about this mechanism]

9.3. New Registry for Notification-Capability Parameters

IANA has created a new registry for the notification-capability parameters of the notify_method_capability test. This registry contains both vendor-controlled notification-capability values (beginning with "vnd.") and IETF-controlled notification-capability values. Vendor-controlled notification-capability values have the format as defined in the following paragraph and may be registered on a "First Come First Served" basis [IANA-GUIDELINES], by applying to IANA with the form specified later in this section. Registration of

notification-capability values that do not begin with "vnd." are registered using the "Specification Required" policy [IANA-GUIDELINES].

Vendor-controlled notification-capability values MUST have the form "vnd.<vendor-name>.<capability-name>", where <vendor-name> is as specified in the ACAP Vendor Subtree registry [ACAP].

The following template must be used for registering notification-capability parameters:

To: iana@iana.org
Subject: Registration of a new notification-capability parameter
Capability name: [the name of the notification-capability]
Description: [an explanation of the purpose of the notification-capability]
Syntax: [formal definition of allowed values and their syntax]
Permanent and readily available reference(s): [the RFC number(s) or an URL of the document that defines this notification mechanism]
Contact information: [the name and email address of the technical contact for information about this mechanism]

Below is the registration form for the "online" notification-capability:

To: iana@iana.org
Subject: Registration of a new notification-capability parameter
Capability name: online
Description: Returns whether the entity identified by the notification-uri parameter to the notify_method_capability test can receive a notify notification immediately.
Syntax: Can contain one of three values: "yes", "no", and, "maybe". Values MUST be in lowercase.
Permanent and readily available reference(s): This RFC
Contact information: The Sieve discussion list <ietf-mta-filters@imc.org>

10. Acknowledgements

Thanks to Larry Greenfield, Sarah Robeson, Tim Showalter, Cyrus Daboo, Nigel Swinson, Kjetil Torgrim Homme, Michael Haardt, Mark E. Mallett, Ned Freed, Lisa Dusseault, Dilyan Palauzov, Arnt Gulbrandsen, Peter Saint-Andre, Sean Turner, Cullen Jennings, and Pasi Eronen for help with this document.

11. References

11.1. Normative References

[ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

[MailTo] Leiba, B. and M. Haardt, "Sieve Notification Mechanism: mailto", RFC 5436, January 2009.

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

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

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

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

11.2. Informative References

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

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

[TEL-URI] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004.

[XMPP] Saint-Andre, Ed., P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.

[XMPP-URI] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)", RFC 5122, February 2008.

Authors' Addresses

Alexey Melnikov (editor)
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

EMail: Alexey.Melnikov@isode.com

Barry Leiba (editor)
IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
US

Phone: +1 914 784 7941
EMail: leiba@watson.ibm.com

Wolfgang Segmuller
IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
US

Phone: +1 914 784 7408
EMail: werewolf@us.ibm.com

Tim Martin
Endless Crossword
672 Haight st.
San Francisco, CA 94117
US

Phone: +1 510 260-4175
EMail: timmartin@alumni.cmu.edu