

Network Working Group

Updates: [3028](#)

Document: [draft-ietf-sieve-variables-08.txt](#)

Expires Jun 18, 2006

K. T. Homme

University of Oslo

18 Dec 2005

## Sieve Extension: Variables

### Status of this Memo

This document is an Internet-Draft and is subject to all provisions of [section 3 of RFC 3978](#). 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/lid-abstracts.txt>.

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

Distribution of this memo is unlimited.

### Abstract

In advanced mail filtering rule sets, it is useful to keep state or configuration details across rules. This document updates the Sieve filtering language ([RFC 3028](#)) with an extension to support variables. The extension changes the interpretation of strings, adds an action to store data in variables, and supplies a new test so that the value of a string can be examined.

Internet Draft

Sieve Extension: Variables

18 Dec 2005

## 0. Meta-information on this draft

This information is intended to facilitate discussion. It will be removed when this document leaves the Internet-Draft stage.

### 0.1. Discussion

This draft is intended to be an extension to the Sieve mail filtering language, available from the RFC repository as [<http://ftp.ietf.org/rfc/rfc3028.txt>](http://ftp.ietf.org/rfc/rfc3028.txt).

This draft and the Sieve language itself are being discussed on the MTA Filters mailing list at [<ietf-mta-filters@imc.org>](mailto:ietf-mta-filters@imc.org). Subscription requests can be sent to [<ietf-mta-filters-request@imc.org>](mailto:ietf-mta-filters-request@imc.org) (send an mail message with the word "subscribe" in the body). More information on the mailing list along with a WWW archive of back messages is available at [<http://www.imc.org/ietf-mta-filters/>](http://www.imc.org/ietf-mta-filters/).

### 0.2. Noted Changes

#### 0.2.1. Changes since -00

- a) allow generic time zone names, without requiring implementations to support it. added a "\${timezone}" variable so that the user can check if the implementation does support the time zone name he wants. the default time zone was changed to localtime again.
- b) allow back references from :matches as well as :regex.
- c) added a section on implementation limits.
- d) clarified global scope so that it spans include.
- e) clarified that this draft only affects scripts which require "variables".
- f) changed modifiers into being tagged arguments for SET, added precedence table.
- g) added optional COMPARATOR to SET to solve the internationalisation problem with :lower etc.

- h) the name of the variable being SET is passed in a string to conform with overall Sieve grammar. this string is explicitly disallowed from containing variable references.

#### [0.2.2.](#) Changes since -01

- a) clarify that a character is a Unicode character.
- b) added paragraph warning against relying on Sieve for virus checking to security section.
- c) added a paragraph defining constant string.
- d) added namespace to grammar.
- e) removed SETDATE.
- f) added wording and example requiring short-circuiting of test evaluation.

#### [0.2.3.](#) Changes since -02

- a) add references to Unicode and UTF-8, also more boilerplate
- b) fixed a meaningless example.
- c) changed term "numeric variables" to "numbered variables" to reduce the chance of it being interpreted as variables holding integer values.
- d) allow future extensions to access the raw string value.
- e) an unsuccessful match does NOT reset the numbered variables.
- f) added definition of "string :count"
- g) exceeding implementation limits on variable lengths should not make scripts abort.

#### [0.2.4.](#) Changes since -03

- a) clarify short-circuiting.
- b) editorial changes.

#### [0.2.5.](#) Changes since -04

- a) the wildcards in :matches was changed from greedy to non-greedy to better support "principle of least surprise". added example to

Homme

[Page 3]

---

Internet Draft

Sieve Extension: Variables

18 Dec 2005

illustrate the difference.

- b) add definition of "variable"; clarify grammar is based on [[SIEVE](#)]; clarify role of namespaces; add informative references for [[REGEX](#)] and [[SPAMTEST](#)]; add normative reference for [[RELATIONAL](#)]
- c) the use of unsupported numbered variables must be flagged as a syntax error by implementations.

#### [0.2.6.](#) Changes since -00 (WG series)

- a) added example for string test
- b) moved introductory text for MODIFIER from 5.1 into 5.0
- c) added Syntax line for MODIFIER.
- d) added comment to an example showing that the non-greedy "\*" still matches everything due to implicit anchors.
- e) added example of expansion of string with unbalanced braces.
- f) updated reference to [[SPAMTEST](#)].

#### [0.2.7.](#) Changes since -01

- a) moved References from appendix into the document itself.

- b) added example of SET with a comparator.
- c) changed "highest value" to the less ambiguous "largest value".
- d) updated reference to [\[UTF-8\]](#).
- e) allow numbered variables in namespaces.
- f) change `${0}` to mean the complete match.

#### [0.2.8.](#) Changes since -02

- a) explicitly state compatibility with actions in base spec.
- b) "numbered variables" are now called "match variables".

- c) clarify definition of "match variable".
- d) it's not the whole namespace which should match the extension keyword, only the first component.
- e) allow level 2 and above of the namespace specification to be all-digit.
- f) combining `:upper` and `:lower` etc. is now a syntax error.
- g) allow SET to set variables in namespaces if the extension allows it.

#### [0.2.9.](#) Changes since -03

- a) added two new modifiers, `:quoteregex` and `:quotewildcard`.
- b) added wording about security implications of silent truncation.

#### [0.2.10.](#) Changes since -04

- a) fix buggy markup and add missing modifier to syntax description

- b) changed two "syntax error" (which really weren't) into just "error".
- c) changed "Syntax:" into "Usage:" to mirror [[SIEVE](#)] convention.
- d) removed description of regex interaction and :quoteregex
- e) added note to clarify that `${0010}` is the same as `${10}`.
- f) changed name of document to align better with other extensions (uses same format at 3431 and 3894)

[0.2.11](#). Changes since -05

- a) removed "open issues" section.
- b) updated [[RELATIONAL](#)] reference

[0.2.12](#). Changes since -06

- a) updated abstract to mention what this document extends.
- b) changed default scoping behaviour in anticipation of "include" extension.

- c) updated reference to [RFC 2234](#).
- d) clarified whitespace stripping behaviour for "string" test.

[0.2.13](#). Changes since -07

- a) Replaced reference to Unicode with reference to ISO 10646 and made it informational rather than normative.
- b) Updated [[ABNF](#)] since it has been published.
- c) Removed the use of comparator with SET to affect case folding. Restrict case modifiers to US ASCII.
- d) Mention in abstract that this draft updates [RFC 3028](#).

- e) Clarify that match variables contain unmodified extracts from the source value.
- f) Include "INBOX" in all mailbox names for consistency.

## 1. Introduction

This is an extension to the Sieve language defined by [[SIEVE](#)]. It adds support for storing and referencing named data. The mechanisms detailed in this document will only apply to Sieve scripts that include a require clause for the "variables" extension. The require clauses themselves are not affected by this extension.

Conventions for notations are as in [[SIEVE](#)] [section 1.1](#), including use of [[KEYWORDS](#)] and [[ABNF](#)]. The grammar builds on the grammar of [[SIEVE](#)]. In this document, "character" means a character from the ISO 10646 coded character set [[ISO10646](#)], which may consist of multiple octets coded in [[UTF-8](#)], and "variable" is a named reference to data stored or read back using the mechanisms of this extension.

## 2. Capability Identifier

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

## 3. Interpretation of strings

This extension changes the semantics of quoted-string, multi-line-literal and multi-line-dotstuff found in [[SIEVE](#)] to enable the

inclusion of the value of variables.

When a string is evaluated, substrings matching variable-ref SHALL be replaced by the value of variable-name. Only one pass through the string SHALL be done. Variable names are case insensitive, so "foo" and "FOO" refer to the same variable. Unknown variables are replaced by the empty string.

variable-ref	=	"\${" [namespace] variable-name "}"
namespace	=	identifier "." *sub-namespace
sub-namespace	=	variable-name "."
variable-name	=	num-variable / identifier
num-variable	=	1*DIGIT

Examples:

"&\${}!"	=> unchanged, as the empty string is an illegal identifier
"\${doh!}"	=> unchanged, as "!" is illegal in identifiers

The variable "company" holds the value "ACME". No other variables are set.

"\${full}"	=> the empty string
"\${company}"	=> "ACME"
"\${BAD\${Company}}"	=> "\${BADACME}"
"\${President, \${Company} Inc.}"	=> "\${President, ACME Inc.}"

The expanded string MUST use the variable values which are current when control reaches the statement the string is part of.

Strings where no variable substitutions take place are referred to as constant strings. Future extensions may specify that passing non-constant strings as arguments to its actions or tests is an error.

Namespaces are meant for future extensions which make internal state available through variables. These variables SHOULD be put in a namespace whose first component is the same as its capability string. Such extensions SHOULD state which, if any, of the variables in its namespace are modifiable with the "set" action.

References to namespaces without a prior require statement for the relevant extension MUST cause an error.

Tests or actions in future extensions may need to access the unexpanded version of the string argument and, e.g., do the expansion after setting variables in its namespace. The design of the implementation should allow this.



The semantics of quoting using backslash are not changed: backslash quoting is resolved before doing variable substitution.

Examples:

```
"${fo\o}" => ${foo} => the expansion of variable foo.  
"${fo\\o}" => ${fo\o} => illegal identifier => left verbatim.  
"\${foo}" => ${foo} => the expansion of variable foo.  
"\\${foo}" => \${foo} => a backslash character followed by the  
                        expansion of variable foo.
```

If it is required to include a character sequence such as "\${beep}" verbatim in a text literal, the user can define a variable to circumvent expansion to the empty string.

Example:

```
set "dollar" "$";  
set "text" "regarding ${dollar}{beep}";
```

### [3.2.](#) Match variables

A "match variable" has a name consisting only of decimal digits and has no namespace component.

The decimal value of the match variable name will index the list of matching strings from the most recently evaluated successful match of type "matches". The list is empty if no match has been successful.

Note: Extra leading zeroes are allowed and ignored.

The list will contain one string for each wildcard ("?" and "\*") in the match pattern. Each string holds the substring from the source value that the corresponding wildcard expands to, possibly the empty string. The wildcards match as little as possible (non-greedy matching).

The first string in the list has index 1. If the index is out of range, the empty string will be substituted. Index 0 contains the matched part of the source value.

The interpreter MUST short-circuit tests, ie. not perform more tests than necessary to find the result. Evaluation order MUST be left to right. If a test has two or more list arguments, the implementation is free to choose which to iterate over first.

An extension describing a new match type (e.g., [[REGEX](#)]) MAY specify

that match variables are set as a side effect when the match type is used in a script which has enabled the "variables" extension.

Example:

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

if header :matches "List-ID" "*<*@*" {
    fileinto "INBOX.lists.${2}"; stop;
}

# Imagine the header
# Subject: [acme-users] [fwd] version 1.0 is out
if header :matches "Subject" "[*] *" {
    # ${1} will hold "acme-users",
    # ${2} will hold "[fwd] version 1.0 is out"
    fileinfo "INBOX.lists.${1}"; stop;
}

# Imagine the header
# To: coyote@ACME.Example.COM
if address :matches ["To", "Cc"] ["coyote@*.com",
    "wile@*.com"] {
    # ${0} is the matching address
    # ${1} is always the empty string
    # ${2} is part of the domain name ("ACME.Example")
    fileinto "INBOX.business.${2}"; stop;
} else {
    # Control wouldn't reach this block if any match was
    # successful, so no match variables are set at this
    # point.
}

if anyof (true, address :domain :matches "To" "*.com") {
    # The second test is never evaluated, so there are
    # still no match variables set.
    stop;
}
```

#### [4.](#) Action set

Usage:     set [MODIFIER] <name: string> <value: string>

The "set" action stores the specified value in the variable identified by name. The name MUST be a constant string and conform

to the syntax of variable-name. Match variables can not be set. A namespace can not be used unless an extension explicitly allows its

use in "set". An invalid name MUST be detected as a syntax error.

Modifiers are applied on a value before it is stored in the variable. See next section for details.

Variables are only visible to the currently running script. Note: Future extensions may provide different scoping rules for variables.

Variable names are case insensitive.

Example:

```
set "honorific" "Mr";
set "first_name" "Wile";
set "last_name" "Coyote";
set "vacation" text:
Dear ${HONORIFIC} ${last_name},
I'm out, please leave a message after the beep.
.
;
```

"set" does not affect the implicit keep. It is compatible with all actions defined in [[SIEVE](#)].

#### [4.1](#). Modifiers

Usage:     ":lower" / ":upper" / ":lowerfirst" / ":upperfirst" /  
          ":quotewildcard" / ":length"

Modifier names are case insensitive. Unknown modifiers MUST yield a syntax error. More than one modifier can be specified, in which case they are applied according to this precedence list, largest value first:

Precedence	Modifier
40	:lower :upper
30	:lowerfirst :upperfirst
20	:quotewildcard
10	:length

It is an error to use two or more modifiers of the same precedence in a single "set" action.

Examples:

```
# The value assigned to the variable is printed after the arrow
set "a" "juMBlEd lETtERs";           => "juMBlEd lETtERs"
set :length "b" "${a}";               => "15"
set :lower "b" "${a}";                => "jumbled letters"
set :upperfirst "b" "${a}";           => "JuMBlEd lETtERs"
set :upperfirst :lower "b" "${a}";    => "Jumbled letters"
set :quotewildcard "b" "Rock*";       => "Rock\*"
```

#### [4.1.1.](#) Modifier ":length"

The value is the decimal number of characters in the expansion, converted to a string.

#### [4.1.2.](#) Modifier `:quotewildcard`

This modifier adds the necessary quoting to ensure that the expanded text will only match a literal occurrence if used as a parameter to `:matches`. Every character with special meaning ("`*`", "`?`" and "`\`") is prefixed with "`\`" in the expansion.

#### [4.1.3.](#) Case modifiers

These modifiers change the letters of the text from upper to lower case or vice versa. Characters other than "`A`"-"`Z`", "`a`"-"`z`" from US-ASCII are left unchanged.

##### [4.1.3.1.](#) Modifier `:upper`

All lower case letters are converted to their upper case counterpart.

##### [4.1.3.2.](#) Modifier `:lower`

All upper case letters are converted to their lower case counterpart.

##### [4.1.3.3.](#) Modifier `:upperfirst`

The first character of the string is converted to upper case if it is a letter and set in lower case. The rest of the string is left unchanged.

##### [4.1.3.4.](#) Modifier `:lowerfirst`

The first character of the string is converted to lower case if it is a letter and set in upper case. The rest of the string is left unchanged.

## [5.](#) Test string

Usage:     string [MATCH-TYPE] [COMPARATOR]  
          <source: string-list> <key-list: string-list>

The "string" test evaluates to true if any of the source strings matches any key. The type of match defaults to ":is".

In the "string" test, both source and key-list are taken from the script, not the message, and whitespace stripping MUST NOT be done unless the script explicitly requests this through some future mechanism.

Example:

```
set "state" "${state} pending";
if string :matches " ${state} " "* pending *" {
    # the above test always succeeds
}
```

The "relational" extension [[RELATIONAL](#)] adds a match type called ":count". The count of a single string is 0 if it is the empty string, or 1 otherwise. The count of a string list is the sum of the counts of the member strings.

## [6.](#) Implementation Limits

An implementation of this draft MUST support at least 128 distinct variables. The supported length of variable names MUST be at least 32 characters. Each variable MUST be able to hold at least 4000 characters. Attempts to set the variable to a value larger than what the implementation supports SHOULD be reported as an error at compile-time if possible. If the attempt is discovered during run-time, the value SHOULD be truncated and it MUST NOT be treated as an error.

Match variables \${1} through \${9} MUST be supported. References to higher indices than the implementation supports MUST be treated as a syntax error which SHOULD be discovered at compile-time.

## [7.](#) Security Considerations

When match variables are used, and the author of the script isn't careful, strings can contain arbitrary values controlled by the sender of the mail.

Since values stored by "set" which exceed implementation limits are silently truncated, it's not appropriate to store large structures with security implications in variables.

The introduction of variables makes advanced decision making easier to write, but since no looping construct is provided, all Sieve scripts will terminate in an orderly manner.

Sieve filtering should not be relied on as a security measure against hostile mail messages. Sieve is designed to do simple, mostly static tests, and is not suitable for use as a spam or virus checker, where the perpetrator has a motivation to vary the format of the mail in order to avoid filtering rules. See also [[SPAMTEST](#)].

## [8.](#) IANA Considerations

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

To: [iana@iana.org](mailto:iana@iana.org)  
Subject: Registration of new Sieve extension

Capability name: variables  
Capability keyword: variables  
Capability arguments: N/A

Standards Track/IESG-approved experimental RFC number:  
this RFC

Person and email address to contact for further information:  
Kjetil Torgrim Homme  
[kjetilho@ifi.uio.no](mailto:kjetilho@ifi.uio.no)

This information should be added to the list of sieve extensions given on <http://www.iana.org/assignments/sieve-extensions>.

## [9.](#) Acknowledgments

Thanks to Cyrus Daboo, Jutta Degener, Ned Freed, Lawrence Greenfield, Jeffrey Hutzelman, Mark E. Mallett, Alexey Melnikov, Peder Stray and Nigel Swinson for valuable feedback.

## 10. Author's Address

Kjetil T. Homme  
University of Oslo  
PO Box 1080  
0316 Oslo, Norway

Phone: +47 9366 0091  
E-mail: kjetilho@ifi.uio.no

## 11. References

### 11.1. Normative references

- [ABNF] Crocker, D. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", [RFC 4234](#), October 2005.
- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.
- [RELATIONAL] Leiba, B. and Segmuller, W., "Sieve Extension: Relational Tests", Work in Progress, [draft-ietf-sieve-3431bis-XX.txt](#)
- [SIEVE] Guenther, P. and Showalter, T., "Sieve: An Email Filtering Language", Work in Progress, [draft-ietf-sieve-3028bis-XX.txt](#)

- [UTF-8] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", [RFC 3629](#), November 2003.



## [11.2.](#) Informative References

- [ISO10646] ISO/IEC, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", May 1993, with amendments.
- [REGEX] Murchison, K., "Sieve Email Filtering -- Regular Expression Extension", Work in Progress.
- [SPAMTEST] Daboo, C., "SIEVE Email Filtering: Spamtest and VirusTest Extensions", [RFC 3685](#), February 2004

## [Appendix B.](#) Intellectual Property Rights Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in [BCP-11](#). Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

## [Appendix C.](#) Full Copyright Statement

Copyright (C) The Internet Society (2005).

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 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).

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

