

Network Working Group	C. Daboo	
Internet-Draft	A. Stone	
Intended status: Standards Track	July 13, 2010	
Expires: January 14, 2011		

[TOC](#)

Sieve Email Filtering: Include Extension **draft-ietf-sieve-include-05**

Abstract

The Sieve Email Filtering "include" extension permits users to include one Sieve script inside another. This can make managing large scripts or multiple sets of scripts much easier, and allows a site and its users to build up libraries of scripts. Users are able to include their own personal scripts or site-wide scripts.

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 January 14, 2011.

Copyright Notice

Copyright (c) 2010 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material

may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

- [1.](#) Introduction and Overview
- [2.](#) Conventions Used in This Document
- [3.](#) Include Extension
 - [3.1.](#) General Considerations
 - [3.2.](#) Control Structure include
 - [3.3.](#) Control Structure return
 - [3.4.](#) Interaction with Variables
 - [3.4.1.](#) Control Structure global
 - [3.4.2.](#) Variables Namespace global
- [4.](#) Security Considerations
- [5.](#) IANA Considerations
 - [5.1.](#) "include" Extension Registration
- [6.](#) References
 - [6.1.](#) Normative References
 - [6.2.](#) Informative References
- [Appendix A.](#) Acknowledgments
- [S](#) Authors' Addresses

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

Changes from ietf-04 to ietf-05:

- a. Integrate review from Barry Leiba.

Changes from ietf-03 to ietf-04:

- a. No changes.

Changes from ietf-02 to ietf-03:

- a. Setting a variable then calling `global` on it is an error (something like `'use strict'`).
- b. Specify that the `'global'` keyword is only available when `'variables'` has also been required.
- c. Uploading a script that includes a nonexistent script is not an error at upload time.

Changes from `ietf-01` to `ietf-02`:

- a. Require that script names must be constant strings, not subject to variable expansion.
- b. Try the phrase `immediate script` instead of `current script`.
- c. Clarify that `"global 'varname'"` and `"global.varname"` refer to the same variable.
- d. Drop the requirement the `global` keywords come after `require` and before anything else.

Changes from `ietf-00` to `ietf-01`:

- a. Replaced `import/export` with `global`.
- b. Added `:once` modifier to `include`.
- c. Added `global` namespace to see if it holds water.

Changes from `daboo-06` to `ietf-00`:

- a. None

Changes from `-05` to `-06`:

- a. Aaron Stone joins as author.
- b. Removed `|` characters from the script examples.
- c. Updated draft references to published RFCs.

Changes from `-04` to `-05`:

- a. Fixed examples.
- b. Relaxed requirement that imported/exported variables be set before being used.

Changes from `-03` to `-04`:

- a. Fixed missing 2119 definitions.
- b. Defined interaction with variables through use of import and export commands.

Changes from -02 to -03:

- a. Refreshing expired draft (updated for nits).
- b. Syntax -> Usage.
- c. Updated to 3028bis reference.

Changes from -01 to -02:

- a. Minor formatting changes only - refreshing expired draft.

Changes from -00 to -01:

- a. Added IPR boiler plate.
- b. Re-ordered sections at start to conform to RFC style.
- c. Moved recursion comment into General Considerations section.
- d. Switched to using optional parameter to indicate personal vs global.
- e. Explicitly state that an error occurs when a missing script is included.

1. Introduction and Overview

[TOC](#)

It's convenient to be able to break [SIEVE \(Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language," January 2008.\)](#) [RFC5228] scripts down into smaller components which can be reused in a variety of different circumstances. For example, users may want to have a default script and a special 'vacation' script, the latter being activated when the user goes on vacation. In that case the default actions should continue to be run, but a vacation command should be executed first. One option is to edit the default script to add or remove the vacation command as needed. Another is to have a vacation script that simply has a vacation command and then includes the default script.

2. Conventions Used in This Document

[TOC](#)

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\] \(Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.\)](#).

Conventions for notations are as in [SIEVE \(Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language," January 2008.\) \[RFC5228\]](#) Section 1.1.

The following key phrases are used to describe scripts and script execution:

script a valid Sieve script.

script execution an instance of a Sieve interpreter invoked for a given message delivery, starting with the user's active script and continuing through any included scripts until the message is delivered.

immediate script the individual Sieve script file being executed.

including script the individual Sieve script file that had an include statement which included the immediate script.

3. Include Extension

[TOC](#)

3.1. General Considerations

[TOC](#)

Sieve implementations that implement the "include", "return", and "global" commands described below have an identifier of "include" for use with the capability mechanism. If any of the "include", "return", or "global" commands are used in a script, the "include" capability MUST be listed in the "require" statement in that script.

Sieve implementations need to track the use of actions in included scripts so that implicit "keep" behavior can be properly determined based on whether any actions have executed in any script.

Sieve implementations are allowed to limit the total number of nested included scripts, but MUST provide for a total of at least three levels of nested scripts including the top-level script. An error MUST be generated either when the script is uploaded to the Sieve repository,

or when the script is executed, if any nesting limit is exceeded. If such an error is detected whilst processing a Sieve script, an implicit "keep" action MUST be executed to prevent loss of any messages. Sieve implementations MUST ensure that recursive includes are not possible. For example, if script "A" includes script "B", and script "B" includes script "A" an error MUST be generated either when the script is uploaded to the Sieve repository, or when the script is executed. If such an error is detected whilst processing a Sieve script, an implicit "keep" action MUST be executed to prevent loss of any messages.

Sieve implementations MUST generate an error at execution time if an included script does not exist. Implementations MUST NOT generate errors for scripts missing at upload time, as this would force an upload ordering requirement upon script authors / generators.

If the Sieve "variables" extension [\[RFC5229\] \(Homme, K., "Sieve Email Filtering: Variables Extension," January 2008.\)](#) is present, an issue arises with the "scope" of variables defined in scripts that may include each other. For example, if a script defines the variable "\${status}" with one particular meaning or usage, and another defines "\${status}" with a different meaning, then if one script includes the other there is an issue as to which "\${status}" is being referenced. To solve this problem, Sieve implementations MUST follow the scoping rules defined in [Section 3.4 \(Interaction with Variables\)](#) and support the "global" command defined there.

3.2. Control Structure include

[TOC](#)

Usage: include [LOCATION] [ONCE] <value: string>

LOCATION = ":personal" / ":global"

ONCE = ":once"

The "include" command takes an optional "location" parameter, an optional ":once" parameter, and a single string argument representing the name of the script to include for processing at that point. It is RECOMMENDED that implementations restrict script names according to [\[I-D.ietf-sieve-managesieve\] \(Melnikov, A. and T. Martin, "A Protocol for Remotely Managing Sieve Scripts," January 2009.\)](#) Section 1.7.

Implementations MUST NOT allow variables to be expanded into the names of Sieve scripts; in other words, the value MUST be a constant string as defined in [VARIABLES \(Homme, K., "Sieve Email Filtering: Variables Extension," January 2008.\)](#) [RFC5229], Section 3.

The "location" parameter MUST default to ":personal" if not specified. The "location" has the following meanings:

:personal

Indicates that the named script is stored in the user's own personal (private) Sieve repository.

:global Indicates that the named script is stored in a site-wide Sieve repository, accessible to all users of the Sieve system.

The ":once" parameter tells the interpreter only to include the named script if it has not already been included at any other point during script execution. If the script has already been included, processing continues immediately following the include command. Implementations MUST NOT generate an error if an "include :once" command names a script whose inclusion would be recursive; in this case, the script MUST be considered previously included and therefore "include :once" will not include it again.

Note: It is RECOMMENDED that script authors / generators use this parameter only when including a script that performs general duties such as declaring global variables and making sanity checks of the environment.

The included script MUST be a valid Sieve script, including having necessary "require" statements for all optional capabilities used by the script. The scope of a "require" statement in an included script is for the immediate script only, not the including script. For example, if script "A" includes script "B", and script "B" uses the "fileinto" extension, script "B" must have a "require" statement for "fileinto", irrespective of whether script "A" has one. In addition, if script "A" does not have a "require" statement for "fileinto", "fileinto" cannot be used anywhere in script "A", even after inclusion of script "B". A "stop" command in an included script MUST stop all script processing, including the processing of the scripts that include the immediate one. The "return" command (described below) stops processing of the immediate script only, and allows the scripts that include it to continue.

Examples:

The user has four scripts stored in their personal repository:

"default"

This is the default active script that includes several others.

```
require ["include"];

include :personal "always_allow";
include :global "spam_tests";
include :personal "spam_tests";
include :personal "mailing_lists";
```

Personal script "always_allow"

This script special-cases some correspondent email addresses and makes sure any message containing those addresses are always kept.

```
if header :is "From" "boss@example.com"
{
    keep;
}
elsif header :is "From" "ceo@example.com"
{
    keep;
}
```

Personal script "spam_tests"

This script does some user-specific spam tests to catch spam messages not caught by the site-wide spam tests.

```
require ["reject"];

if header :contains "Subject" "XXXX"
{
    reject;
}
elsif header :is "From" "money@example.com"
{
    reject;
}
```

Personal script "mailing_lists"

This script looks for messages from different mailing lists and files each into a mailbox specific to the mailing list.

```
require ["fileinto"];

if header :is "List-ID" "owner-ietf-mta-filters@imc.org"
{
    fileinto "lists.sieve";
}
elsif header :is "List-ID" "owner-ietf-imapext@imc.org"
{
    fileinto "lists.imapext";
}
```

There is one script stored in the global repository:
Site script "spam_tests"

This script does some site-wide spam tests which any user at the site can include in their own scripts at a suitable point. The script content is kept up to date by the site administrator.

```
require ["reject"];

if anyof (header :contains "Subject" "$$",
          header :contains "Subject" "Make money")
{
    reject;
}
```

The "include" command may appear anywhere in the script where a control structure is legal.

Example:

```
require ["include"];

if anyof (header :contains "Subject" "$$",
          header :contains "Subject" "Make money")
{
    include "my_reject_script";
}
```

3.3. Control Structure return

[TOC](#)

Usage: return

The "return" command stops processing of the immediately included script only and returns processing control to the script which includes it. If used in the main script (i.e., not in an included script), it has the same effect as the "stop" command, including the appropriate "keep" action if no other actions have been executed up to that point.

3.4. Interaction with Variables

[TOC](#)

In order to avoid problems of variables in an included script "overwriting" those from the script that includes it, this specification requires that all variables defined in a script MUST be kept "private" to the immediate script by default - that is, they are not "visible" to other scripts. This ensures that two script authors

cannot inadvertently cause problems by choosing the same name for a variable.

However, sometimes there is a need to make a variable defined in one script available to others. This specification defines the new command "global" to declare that a variable is shared among scripts.

Effectively, two namespaces are defined: one local to the immediate script, and another shared among all scripts. Implementations **MUST** allow a non-global variable to have the same name as a global variable but have no interaction between them.

3.4.1. Control Structure global

[TOC](#)

Usage: global <value: string-list>

The "global" command contains a string list argument that defines one or more names of variables to be stored in the global variable space. The "global" command is only available when the script has both "include" and "variables" in its require line. If the "global" command appears when only "include" or only "variables" has been required, an error **MUST** be generated when the script is uploaded.

If a "global" command is given the name of a variable that has previously been defined in the immediate script with "set", an error **MUST** be generated either when the script is uploaded or at execution time.

If a "global" command lists a variable that has not been defined in the global namespace, the name of the variable is now marked as global, and any subsequent "set" command will set the value of the variable in global scope.

A variable has global scope in all scripts that have declared it with the "global" command. If a script uses that variable name without declaring it global, the name specifies a separate, non-global variable within that script.

Interpretation of a string containing a variable marked as global, but without any value set, **SHALL** behave as any other access to an unknown variable, as specified in [VARIABLES \(Homme, K., "Sieve Email Filtering: Variables Extension," January 2008.\)](#) [RFC5229], Section 3 (i.e., evaluates to an empty string).

Example:

```

require ["variables", "include"];
global "test";
global "test-mailbox";

# The included script may contain repetitive code that is
# effectively a subroutine that can be factored out.
set "test" "$$"
include "spam_filter_script";

set "test" "Make money"
include "spam_filter_script";

# Message will be filed according to the test that matched last.
if string :count "${test-mailbox}" "1"
{
    fileinto "INBOX${test-mailbox}";
    stop;
}

# If nothing matched, the message is implicitly kept.

```

Active script

```

require ["variables", "include"];
global ["test", "test-mailbox"];

if header :contains "Subject" "${test}"
{
    set "test-mailbox" "spam-${test}";
}

```

spam_filter_script

3.4.2. Variables Namespace global

[TOC](#)

In addition to the "global" command, this document defines the variables namespace "global", as specified in [VARIABLES \(Homme, K.,](#)

["Sieve Email Filtering: Variables Extension," January 2008.](#)) [RFC5229],
Section 3.
Example:

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

set "global.i_am_on_vacation" "1";
```

Variables declared global and variables accessed via the global namespace MUST be one and the same. In the following example script, we see the variable "i_am_on_vacation" used in a "global" command, and again with the "global." namespace. Consider these as two syntaxes with identical meaning.

Example:

```
require ["variables", "include"];
global "i_am_on_vacation";

set "global.i_am_on_vacation" "1";

if string :is "${i_am_on_vacation}" "1"
{
    vacation "It's true, I am on vacation."
}
```

4. Security Considerations

[TOC](#)

Sieve implementations MUST ensure adequate security for the global script repository to prevent unauthorized changes to global scripts. Sieve implementations MUST ensure that script names are checked for validity and proper permissions prior to inclusion, in order to prevent a malicious user from gaining access to files accessible to the mail server software that should not be accessible to the user. Beyond these, the "include" extension does not raise any security considerations that are not present in the base [SIEVE \(Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language," January 2008.\)](#) [RFC5228] document and the [VARIABLES \(Homme, K., "Sieve Email Filtering: Variables Extension," January 2008.\)](#) [RFC5229] extension.

5. IANA Considerations

[TOC](#)

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

5.1. "include" Extension Registration

[TOC](#)

Capability name: include

Description: adds the "include" command to execute other Sieve scripts, and the "global" command and "global" variables namespace to access variables shared among included scripts.

RFC number: this RFC

Contact address: the Sieve discussion list <ietf-mta-filters@imc.org>

6. References

[TOC](#)

6.1. Normative References

[TOC](#)

[RFC2119]	Bradner, S. , " Key words for use in RFCs to Indicate Requirement Levels ," BCP 14, RFC 2119, March 1997 (TXT , HTML , XML).
[RFC5228]	Guenther, P. and T. Showalter, " Sieve: An Email Filtering Language ," RFC 5228, January 2008 (TXT).
[RFC5229]	Homme, K., " Sieve Email Filtering: Variables Extension ," RFC 5229, January 2008 (TXT).

6.2. Informative References

[TOC](#)

[I-D.ietf-sieve-managesieve]	Melnikov, A. and T. Martin, " A Protocol for Remotely Managing Sieve Scripts ," draft-ietf-sieve-managesieve-09 (work in progress), January 2009 (TXT).
------------------------------	---

Appendix A. Acknowledgments

[TOC](#)

Thanks to Ken Murchison, Rob Siemborski, Alexey Melnikov, Marc Mutz, Kjetil Torgrim Homme, Stephan Bosch, Arnt Gulbrandsen, Barry Leiba, and Jeffrey Hutzelman for comments and corrections.

Authors' Addresses[TOC](#)

	Cyrus Daboo
Email:	cyrus@daboo.name
	Aaron Stone
Email:	aaron@serendipity.cx