

Network Working Group
Internet-Draft
Expires: September 24, 2009

C. Daboo
A. Stone
March 23, 2009

Sieve Email Filtering: Include Extension
draft-ietf-sieve-include-00

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#). 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.

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>.

This Internet-Draft will expire on September 24, 2009.

Copyright Notice

Copyright (c) 2009 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 in effect on the date of

Internet-Draft

Sieve Extension: Include

March 2009

publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

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, as well as supporting common 'libraries' of scripts. Users are able to include their own personal scripts or site-wide scripts provided by the local Sieve implementation.

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

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.

Open Issues (to be resolved prior to publication as an RFC)

- a. Interaction with variables (scoping). Should variables be carried over between scripts that are included? Or should variables defined in an included script be local to that script only?

Table of Contents

1.	Introduction and Overview	4
2.	Conventions Used in This Document	4
3.	Include Extension	4
3.1.	General Considerations	4
3.2.	Control Structure Include	5
3.3.	Control Structure Return	8
3.4.	Interaction with Variables	8
3.4.1.	Control Structure Import	8
3.4.2.	Control Structure Export	9
4.	Security Considerations	10
5.	IANA Considerations	10
5.1.	"include" Extension Registration	10
6.	Normative References	11
Appendix A.	Acknowledgments	11
	Authors' Addresses	11

1. Introduction and Overview

Its convenient to be able to break SIEVE [[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 later 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

Conventions for notations are as in SIEVE [[RFC5228](#)] [Section 1.1](#).

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](#)].

3. Include Extension

3.1. General Considerations

Sieve implementations that implement the "include", "return", "export" and "import" commands described below have an identifier of

"include" for use with the capability mechanism. If any of the "include", "return", "export" or "import" commands are used in a script, the "include" capability MUST be listed in the "require" statement in that script.

Sieve implementations must 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. i.e. 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 handle missing scripts being referenced via an includes in an existing script. An error MUST be generated when a missing included script is discovered during execution. If such an error is detected an implicit "keep" action MUST be executed to prevent loss of any messages.

If the Sieve "variables" extension [[RFC5229](#)] 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](#) and support the "import" and "export" commands defined there.

[3.2.](#) Control Structure Include

Usage: `include [<location: LOCATION>] <value: string>`

`LOCATION = ":personal" / ":global"`

The "include" command includes an optional "location" parameter, and a single string argument representing the name of the script to include in the main script at that point.

The "location" parameter defaults 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 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 that script only, not the including script. e.g. 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 current one. The "return" command (described below) stops processing of the current 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 "my_spam_tests";
include :personal "mailing_lists";
```

"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;
}
```

"my_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;
```

```
}
elseif header :is "From" "money@example.com"
{
    reject;
}
```

"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 "Sender" "owner-ietf-mta-filters@imc.org"
{
    fileinto "lists.sieve";
}
elseif header :is "Sender" "owner-ietf-imapext@imc.org"
{
    fileinto "lists.imapext";
}
```

There is one script stored in the global repository:

"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

Usage: return

The "return" command stops processing of the currently 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

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 that script by default - i.e. 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 commands "export" and "import" to alter the default behavior of variable scoping to allow variables to be "seen" by other scripts. The "export" command takes a list of variable names defined in a script and makes those available to any script that explicitly wants to use them. The "import" command allows a script to gain access to variables that have been explicitly made available by other scripts. The explicit use of "export" and "import", coupled with the default behavior of variables only having local scope, ensures that multiple scripts cannot inadvertently overwrite each others variables.

[3.4.1.](#) Control Structure Import

Usage: import <value: string-list>

The "import" command contains a string list argument that defines one

or more names of variables exported by other scripts which should be made available to the current script.

The "import" command, if present, MUST be used immediately after any "require" commands (at least one of which will be present listing the "include" extension). Multiple "import" commands are allowed. An error occurs if an "import" command appears after a command other than "require" or "import". Use of the "import" command makes the listed variables immediately available for use in the body of the script that uses it.

If an "import" command lists a variable that has not been exported by any other script at that point during the Sieve execution process, then an error MUST occur.

Example:

```
require ["variables", "fileinto", "include"];
import "test";

if header :contains "Subject" "${test}"
{
    fileinto "INBOX.spam-${test};
}
```

[3.4.2.](#) Control Structure Export

Usage: export <value: string-list>

The "export" command contains a string list argument that defines one or more names of variables defined in the current script which should be made available to any scripts that run during the current Sieve script execution process.

The "export" command, if present, MUST be used immediately after any "require" or "import" commands. Multiple "export" commands are allowed. An error occurs if an "export" command appears after a command other than "require", "import" or "export".

An error occurs if an "export" command lists a variable that is listed in an "import" command preceding it in the current script.

Example:

Internet-Draft

Sieve Extension: Include

March 2009

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

set "test" "$$"
include "spam_filter_script";

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

Example:

```
require ["variables", "include"];
import "test";
export "test-mailbox";

if header :contains "Subject" "${test}"
{
    set "test-mailbox" "INBOX.spam-${test}";
    include "fileinto-script"
}
```

[4.](#) Security Considerations

Sieve implementations MUST ensure adequate security for the global script repository to prevent unauthorized changes to global scripts.

Beyond that, the "include" extension does not raise any security considerations that are not present in the base Sieve protocol, and these issues are discussed in Sieve.

[5.](#) IANA Considerations

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

[5.1.](#) "include" Extension Registration

Capability name: include
Description: add the "include" command to execute other Sieve scripts.
RFC number: this RFC
Contact address: the Sieve discussion list <ietf-mta-filters@imc.org>

Daboo & Stone Expires September 24, 2009 [Page 10]

Internet-Draft Sieve Extension: Include March 2009

[6.](#) Normative References

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

[Appendix A.](#) Acknowledgments

Thanks to Ken Murchison, Rob Siemborski, Alexey Melnikov, Marc Mutz and Kjetil Torgrim Homme for comments and corrections.

Authors' Addresses

Cyrus Daboo

Email: cyrus@daboo.name

Aaron Stone

Email: aaron@serendipity.palo-alto.ca.us

