Search docs

UPDATE YOUR ACCOUNT
SHARING YOUR PRODUCTS
USING ORGANIZATIONS
DOMAIN NAMES
MARKETPLACE
USING EMAIL
Email Connection Settings
Manage Email Options
Forward Email or Create an Alias
Limitations of Gandi Mailboxes
SOGo Webmail
Roundcube Webmail
Sieve Email Filters
A Basic Introduction to Sieve Filters

Docs » Using Email » How to Use Sieve Filters to Filter Your Email » Write Your Own Sieve Filters Tutorial

Write Your Own Sieve Filters Tutorial

- What's In This Tutorial
- List Extensions Used
- Comments
- Creating Filter Criteria
- Perform Actions on a Filtered Message
- Adding More Criteria
- Sending Emails to a Specific Folder
- Testing for Multiple Criteria at Once
- Send Remaining Email to a Folder

The following is a tutorial showing how to create Sieve scripts using code. If you are brand new to writing Sieve scripts you may want to look at our getting started page first.

What's In This Tutorial

This tutorial will take you through the process of writing a Sieve script with code. It will cover many of the basic elements of using Sieve. You will learn how to create a filter which does the following:

- Reject messages over a size limit
- Filter email from a mailing list into a specific folder
- Filter email with suspicious subjects into the junk mail folder

There are many possible actions with Sieve, many which are not covered in this tutorial. This tutorial will give you an understanding of how Sieve filters are structured so that you are able to put all Sieve commands together into workable filters. You can find a complete list of the extensions Gandi supports here. You can also find some more example filters here.

List Extensions Used

The first step in writing a Sieve filter is listing the required extensions.

The sieve core code is composed of some basic built in commands as well as extensions that add additional capability. The first line in your sieve filter file will be the "**require**" command followed by the list of extensions you will be using in your filters. In the filters covered in this tutorial you will need the "fileinto" and "reject" extensions.

Your first line will look this:

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

You can find a complete list of Sieve extensions supported by Gandi's servers here.

Comments

When writing code, comments refer to text that is ignored when the code is processed. It is a good idea to use comments to explain what you are trying to do with the code so that when you come back later it is easier to edit. It also makes it easier to share your code with others.

You can create a comment in Sieve by using a hashtag to comment the entire contents of a single line, like this:

```
#This is a comment.
```

Or you can create a comment that uses only a partial line, or multiple lines, like this:

```
/* This is also a comment. */

/* This
* is
* a comment
* which spans multiple lines. */
```

The filter you create in this tutorial includes comments explaining each step of the filter. The filter should look like this:

```
# My Sieve Filter

# List the extensions used by this script
require ["fileinto", "reject"];
```

Creating Filter Criteria

The filter starts by blocking any emails that are bigger than 2MB. But first, you need to tell the filter what you are looking for. You do this with an **if** command. Don't forget to add a comment as well. With the comment and if statement, your filter shoudl now look like this:

```
# Messages bigger than 2MB will be rejected with an error message
if
```

Next, you will tell the filter what part of the message to look at. This could be the from address, the to address, the subject, the date sent, etc. This filter will look at the **size** of the email:

```
# Messages bigger than 2MB will be rejected with an error message
if size
```

Next, you want to tell the filter how to compare the criteria you give it with the contents of the message. These are called *comparators* and they are always preceded by a single colon. For this filter you will use **:over**. This tells the filter to look at on messages that are *over*, or bigger, than the size you specify. In this case, the amount should be 2000K since that is equal to 2 MB.

We have now given the filter a full set of criteria to look for. To tell the filter that your test is done use an opening curly brace **{**. The entire filter up to this point will look like this:

```
# My Sieve Filter

# List the extensions used by this script
require ["fileinto", "reject"];

# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
```

Perform Actions on a Filtered Message

After you write what messages you are looking for, you need to tell the filter what to do with the messages it finds. You do this using curly braces **{** and **}**. Any actions you want taken on the emails that fit your criteria should be within these curly braces.

There are many actions you can take on an email including filing into a specific folder, deleting it, forwarding it, and so on. This filter uses **reject** to refuse acceptance of the message. We do this by writing **reject** on the next line, like this:

```
# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
  reject
```

After **reject** you can give an error message that will be sent to whoever sent the message. The line should then end with a semicolon.

> **Tip**
>
> Every line in the action part of the filter needs to end with a semicolon, so it's important not to forget this important detail.

```
# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
  reject "I am unable to accept email larger than 2MB.";
```

After writing all the actions to be taken, you will end with a closing curly brace. That means the entire filter up to this point will look like this:

```
# My Sieve Filter
```

```
# List the extensions used by this script
require ["fileinto", "reject"];

# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
 reject "I am unable to accept email larger than 2MB.";
}
```

At this point you have created a fully functional Sieve script! You could stop here, but there are still more emails to filter so let's keep going.

Adding More Criteria

You most likely will have several types of messages you wish to filter from your email. To add additional criteria you can use **elsif**. This is short for "else if" and tells the filter that this criteria should be applied to any emails that didn't fit the first criteria.

> **Warning**
>
> When using Sieve code **all** of your filters must be in a **single file**. When you upload a new Sieve file to the mail server it will replace any file currently active. In other words, don't start a new file for a new filter, or your original filters will be deactivated.

For the next criteria the filter will move any email that is coming from a mailing list into a folder. You will start by using **elsif**.

```
# Mails from a mailing list will be put into the folder "mailinglist"
elsif
```

The filter needs to look at the **address** part of the message. Since you are looking for an exact match of the entire address, you will use the comparator :is.

```
# Mails from a mailing list will be put into the folder "mailinglist"
elsif address :is
```

Now that you have specified which part of the email to compare against, you need to write what criteria to look for. This filter will look at both the "to" and "from" addresses and apply the actions if either of those email addresses is the "mailinglist@example.com." This is done using square brackets, followed by the address, like this:

```
# Mails from a mailing list will be put into the folder "mailinglist"
elsif address :is ["From", "To"] "mailinglist@example.com"
```

We then end with an opening curly brace, so the entire filter up to this point looks like this:

```
# My Sieve Filter

# List the extensions used by this script
require ["fileinto", "reject"];

# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
 reject "I am unable to accept email larger than 2MB.";
}

# Mails from a mailing list will be put into the folder "mailinglist"
elsif address :is ["From", "To"] "mailinglist@example.com" {
```

Sending Emails to a Specific Folder

One of the most common uses for Sieve filters is to sort emails into specific folders. This can be done using the **fileinto** command. This command sends the message into a specified folder.

This filter will send messages that are to or from mailinglist@example.com into the folder "mailinglist". Since the mailing box folder is inside the inbox folder, use "INBOX/mailinglist."

```
# Mails from a mailing list will be put into the folder "mailinglist"
elsif address :is ["From", "To"] "mailinglist@example.com" {
  fileinto "INBOX/mailinglist";
}
```

The entire filter up to this point will look like this:

```
# My Sieve Filter

# List the extensions used by this script
require ["fileinto", "reject"];

# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
 reject "I am unable to accept email larger than 2MB.";
}

# Mails from a mailing list will be put into the folder "mailinglist"
elsif address :is ["From", "To"] "mailinglist@example.com" {
  fileinto "INBOX/mailinglist";
}
```

Testing for Multiple Criteria at Once

To test for multiple criteria at once you can use **anyof** or **allof**. The command **anyof** will perform the action so long as at least one of the criteria is met. And, **allof** will perform the action only when all of the criteria are met. This filter will use **anyof**, as shown:

```
# If the email isn't directly addressed to me, or if the subject has "free money"
# prince" in it, put it in the spam folder.
elsif anyof
```

You will then put the criteria inside simple parentheses, separated by a comma. The first criteria will look for any email that is NOT addressed to your email address (for example, “myemailaddress@gmail.com”). To show a negative criteria (in this case any messages not addressed to your email) put **not** at the beginning of the criteria. Many emails are addressed to more than one address, so this filter uses **:contains** since the entire “to”, “cc”, or “bcc” fields do not need to be an exact match.

```
# If the email isn't directly addressed to me, or if the subject has "free money"
# prince" in it, put it in the spam folder.
elsif anyof (not address :all :contains ["To", "Cc", "Bcc"] "myemailaddress@examp
```

The next criteria will look at whether some key phrases appear in the subject line. This time you will use the **:matches** comparator. The **:matches** comparator is very similar to **:contains** but allows you to include wildcards. The wildcard * means anything can fill that space. So *free money* searches for any combination of characters so long as “free money” appears somewhere in the text. This is what this criteria will look like on its own:

```
header :matches "Subject" ["*free money*","*Nigerian prince*"]
```

When you put both criteria together the filter will look like this:

```
# If the email isn't directly addressed to me, or if the subject has "free money"
# prince" in it, put it in the spam folder.
elsif anyof (not address :all :contains ["To", "Cc", "Bcc"] "myemailaddress@examp
header :matches "Subject" ["*free money*","*Nigerian prince*"]) {
```

The final step of the filter will be to use **fileinto** to send these emails to the spam folder. With this last part added, the entire filter up to this point will look like this:

```
# My Sieve Filter

# List the extensions used by this script
require ["fileinto", "reject"];

# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
  reject "I am unable to accept email larger than 2MB.";
}

# Mails from a mailing list will be put into the folder "mailinglist"
elsif address :is ["From", "To"] "mailinglist@example.com" {
  fileinto "INBOX/mailinglist";
}
```

```
# If the email isn't directly addressed to me, or if the subject has "free money"
# prince" in it, put it in the spam folder.
elsif anyof (not address :all :contains ["To", "Cc", "Bcc"] "myemailaddress@examp
header :matches "Subject" ["*free money*","*Nigerian prince*"]) {
  fileinto "INBOX/spam";
}
```

Send Remaining Email to a Folder

For the last command in the filter you will send any email that did not fit any of the criteria into the “to sort” folder. Please note, this is not a required part of a Sieve script. The assumption by the sieve program is that any emails not caught in a filter should simply remain in the inbox. This filter is simply showing how you can send remaining messages into a specific folder if that is what you wish to do.

```
# Put the remaining new mail into a folder to sort later.
else {
  fileinto "INBOX/tosort";
}
```

With this final command the filter is now complete.

```
# My Sieve Filter

# List the extensions used by this script
require ["fileinto", "reject"];

# Messages bigger than 2MB will be rejected with an error message
if size :over 2000K {
  reject "I am unable to accept email larger than 2MB.";
}

# Mails from a mailing list will be put into the folder "mailinglist"
elsif address :is ["From", "To"] "mailinglist@example.com" {
  fileinto "INBOX/mailinglist";
}

# If the email isn't directly addressed to me, or if the subject has "free money"
# prince" in it, put it in the spam folder.
elsif anyof (not address :all :contains ["To", "Cc", "Bcc"] "myemailaddress@examp
header :matches "Subject" ["*free money*","*Nigerian prince*"]) {
  fileinto "INBOX/spam";
}

# Put the remaining new mail into a folder to sort later.
else {
  fileinto "INBOX/tosort";
}
```

Hopefully at this point you now have a grasp of the important elements of a Sieve filter. Remember that Sieve is flexible and there are many more options than are listed here. You can find a complete list of the extensions Gandi supports here. You can also find some more example filters here.

← Previous

Next →

© Copyright 2024, Gandi. Built with Sphinx 4.5.0