imapautofiler Documentation

Release 1.12.0

Doug Hellmann

May 07, 2022

Contents:

imapautofiler is a tool for managing messages on an IMAP server based on rules for matching properties such as the recipient or header content. The author uses it to move messages from his "Sent" folder to the appropriate archive folders.

CHAPTER 1

Installing

Install `imapautofiler` with pip under Python 3.5 or greater.

```
$ pip install imapautofiler
```

1.1 System Package Dependencies

`imapautofiler` uses some libraries for which system packages may need to be installed.

1.1.1 Debian

Before installing `imapautofiler`, install the other required packages

```
$ sudo apt-get install build-essential python3-dev \
  libffi-dev libssl-dev libdbus-1-dev libdbus-glib-1-dev \
  gnome-keyring
```

If you are using a virtualenv, you may also need to install `dbus-python`.

```
$ pip install dbus-python
```

CHAPTER 2

Configuring

The application is configured through a YAML file. The default file, `~/.imapautofiler.yml`, is read if no other file is specified on the command line.

2.1 Server Connection

Each configuration file can hold one server specification.

```
server:
  hostname: example.com
  username: my-user@example.com
```

imapautofiler also supports using the keyring module to store and retrieve a password from your system keyring:

```
server:
  hostname: example.com
  username: my-user@example.com
  use_keyring: true
```

In this scenario, you will be asked for the password on first run. The password will be stored in your operating system's secure keyring and reused when running the app. Refer to the keyring documentation for more details about setting up secure password storage.

If you do not want to use the keyring, the connection section can optionally include a password.

```
server:
  hostname: example.com
  username: my-user@example.com
  password: super-secret
```

Warning: Because the password is kept in clear text, this mode of operation is only recommended when the configuration file is kept secure by other means.

If the password is not provided in the configuration file and `use_keyring` is not true, `imapautofiler` will prompt for a value when it tries to connect to the server.

You can also optionally provide the IMAP servers port and a custom CA file. This is helpful if your company uses custom ports and self issued certs.

```
server:
  hostname: example.com
  username: my-user@example.com
  port: 1234
  ca_file: path/to/ca_file.pem
```

Sometimes servers use an SSL/TLS certificate with a common name, which does not match the hostname you are connecting to. Normally, when encountering this situation, `imapautofiler` will abort the connection with an error:

```
imapautofiler: error: certificate error for imap.example.com: hostname 'imap.example.
↪com' doesn't match 'bogus.example.com'
```

You can disable hostname checking for SSL/TLS certs by setting the `check_hostname` option to a false value (any value other than `y`, `yes`, `t`, `true`, `on`, `enabled`, or `1` (case-insensitive) will be regarded as false. The default is true).

```
server:
  hostname: imap.example.com
  username: my-user@example.com
  check_hostname: false
```

Warning: Use at your own risk! Disabling hostname checking is dangerous and makes the connection vulnerable to man-in-the-middle attacks. You should preferably ask the server operator to install a proper certificate instead.

You can disable SSL/TLS connection by setting the `ssl` option to a false value (any value other than `y`,‘‘yes‘‘, `t`, `true`, `on`, `enabled`, or `1` (case-insensitive) will be regarded as false. The default is true).

```
server:
  hostname: imap.example.com
  username: my-user@example.com
  ssl: false
```

Warning: Use at your own risk! Disabling SSL/TLS connection is dangerous. You should preferably ask the server operator to install a proper certificate instead.

2.2 Maildir Location

As an alternative to a server specification, the configuration file can refer to a local directory containing one or more Maildir folders. This is especially useful when combining imapautofiler with offlineimap.

```
maildir: ~/Mail
```

Note: The directory specified should not itself be a Maildir. It must be a regular directory with nested Maildir folders.

2.3 Trash Mailbox

The `trash` action, for discarding messages without deleting them immediately, requires a configuration setting to know the name of the trash mailbox. There is no default value.

```
trash-mailbox: INBOX.Trash
```

2.4 Mailboxes

The mailboxes that imapautofiler should process are listed under `mailboxes`. Each mailbox has a name and a list of rules.

```
mailboxes:
- name: INBOX
  rules: ...
- name: Sent
  rules: ...
```

2.5 Rules

The rules are organized by mailbox, and then listed in order. The first rule that matches a message triggers the associated action, and then processing for that message stops.

2.5.1 TimeLimit Rules

An Time Limit `time-limit` rule is added by specifying the 'age', number of days for the email to "live" in the specified mailbox. If age = 0, the rule is ignored.

```
- time-limit:
    age: 30
```

2.5.2 Header Rules

A `header` rule can match either a complete header value, a substring, or a regular expression against the contents of a specified message header. If a header does not exist, the content is treated as an empty string. The header text and pattern are both converted to lowercase before the comparison is performed.

This example rule matches messages with the string "[pyatl]" in the subject line.

```
- headers:
    - name: "subject"
      substring: "[pyatl]"
  action:
    name: "move"
    dest-mailbox: "INBOX.PyATL"
```

This example rule matches messages for which the "to" header matches the regular expression `notify-.*@disqus.net`.

```
- headers:
    - name: to
      regex: "notify-.*@disqus.net"
  action:
    name: trash
```

This example rule matches messages for which the "Message-Id" header is exactly `<4FF56508-357B-4E73-82DE-458D3EEB2753@example.com>`.

```
- headers:
    - name: to
      value: "<4FF56508-357B-4E73-82DE-458D3EEB2753@example.com>"
  action:
    name: trash
```

2.5.3 Combination Rules

It is frequently useful to be able to apply the same action to messages with different characteristics. For example, if a mailing list ID appears in the subject line or in the `list-id` header. The `or` rule allows nested rules. If any one matches, the combined rule matches and the associated action is triggered.

For example, this rule matches any message where the PyATL meetup mailing list address is in the `to` or `cc` headers.

```
- or:
    rules:
      - headers:
          - name: "to"
            substring: "pyatl-list@meetup.com"
      - headers:
          - name: "cc"
            substring: "pyatl-list@meetup.com"
  action:
    name: "move"
    dest-mailbox: "INBOX.PyATL"
```

For more complicated formulations, the `and` rule allows combining other rules so that they all must match the message before the action is taken.

For example, this rule matches any message sent to the PyATL meetup mailing list address with a subject including the text `"meeting update"`.

```
- and:
    rules:
      - headers:
          - name: "to"
            substring: "pyatl-list@meetup.com"
      - headers:
          - name: "subject"
            substring: "meeting update"
  action:
    name: "move"
    dest-mailbox: "INBOX.PyATL"
```

2.5.4 Recipient Rules

The example presented for `or` rules is a common enough case that it is supported directly using the `recipient` rule. If any header listing a recipient of the message matches the substring or regular expression, the action is triggered.

This example is equivalent to the example for `or`.

```
- recipient:
    substring: "pyatl-list@meetup.com"
  action:
    name: "move"
    dest-mailbox: "INBOX.PyATL"
```

2.6 Actions

Each rule is associated with an *action* to be triggered when the rule matches a message.

2.6.1 Move Action

The `move` action copies the message to a new mailbox and then deletes the version in the source mailbox. This action can be used to automatically file messages.

The example below moves any message sent to the PyATL meetup group mailing list into the mailbox `INBOX.PyATL`.

```
- recipient:
    substring: "pyatl-list@meetup.com"
  action:
    name: "move"
    dest-mailbox: "INBOX.PyATL"
```

The `dest-mailbox` value can contain jinja2 template directives using the headers of the message. For example

```
- recipient:
    substring: "pyatl-list@meetup.com"
  action:
    name: "move"
    dest-mailbox: "INBOX.PyATL.{{ date.year }}"
```

will extract the year value from the date header of the message and insert it into the destination mailbox path.

Header names are always all lower case and - is replaced by _.

Different IMAP servers may use different naming conventions for mailbox hierarchies. Use the `--list-mailboxes` option to the command line program to print a list of all of the mailboxes known to the account.

2.6.2 Sort Action

The `sort` action uses data in a message header to determine the destination mailbox for the message. This action can be used to automatically file messages from mailing lists or other common sources if the corresponding mailbox hierarchy is established. A `sort` action is equivalent to `move` except that the destination is determined dynamically.

The action settings may contain a `header` entry to specify the name of the mail header to examine to find the destination. The default is to use the `to` header.

The action data may contain a `dest-mailbox-regex` entry for parsing the header value to obtain the destination mailbox name. If the regex has one match group, that substring will be used. If the regex has more than one match group, the `dest-mailbox-regex-group` option must specify which group to use (0-based numerical index). The default pattern is `([\w-+]+)@` to match the first part of an email address.

The action data must contain a `dest-mailbox-base` entry with the base name of the destination mailbox. The actual mailbox name will be constructed by appending the value extracted via `dest-mailbox-regex` to the `dest-mailbox-base` value. The `dest-mailbox-base` value should contain the mailbox separator character (usually `.`) if the desired mailbox is a sub-folder of the name given.

The example below sorts messages associated with two mailing lists into separate mailboxes under a parent mailbox `INBOX.ML`. It uses the default regular expression to extract the prefix of the `to` header for each message. Messages to the `python-committers@python.org` mailing list are sorted into `INBOX.ML.python-committers` and messages to the `sphinx-dev@googlegroups.com` list are sorted into `INBOX.ML.sphinx-dev`.

```
- or:
    rules:
      - recipient:
          substring: python-committers@python.org
      - recipient:
          substring: sphinx-dev@googlegroups.com
  action:
    name: sort
    dest-mailbox-base: "INBOX.ML."
```

The `dest-mailbox-base` may include jinja2 template instructions, which are evaluated before the suffix is added to the base. Refer to the description of the `move` action for more details about template evaluation.

2.6.3 Sort Mailing List Action

The `sort-mailing-list` action works like `sort` configured to read the `list-id` header and extract the portion of the ID between < and >. if they are present. If there are no angle brackets in the ID, the entire value is used. As with `sort` the `dest-mailbox-regex` can be specified in the rule to change this behavior.

The example below sorts messages to any mailing list into separate folders under `INBOX.ML`.

```
- is-mailing-list: {}
  action:
    name: sort-mailing-list
    dest-mailbox-base: "INBOX.ML."
```

2.6.4 Trash Action

Moving messages to the "trash can" is a less immediate way of deleting them. Messages in the trash can can typically be recovered until they expire, or until the trash is emptied explicitly.

Using this action requires setting the global `trash-mailbox` option (see *Trash Mailbox*). If the action is triggered and the option is not set, the action reports an error and processing stops.

This example moves messages for which the "to" header matches the regular expression `notify-.*@disqus.net` to the trash mailbox.

```
- headers:
    - name: to
      regex: "notify-.*@disqus.net"
```

(continues on next page)

(continued from previous page)

```
  action:
    name: trash
```

2.6.5 Delete Action

The `delete` action is more immediately destructive. Messages are permanently removed from the mailbox as soon as the mailbox is closed.

This example deletes messages for which the "to" header matches the regular expression `notify-.*@disqus.net`.

```
- headers:
    - name: to
      regex: "notify-.*@disqus.net"
  action:
    name: delete
```

2.6.6 Flag and Unflag

The `flag` action sets the flag of a message.

```
action:
  name: flag
```

The `unflag` action unsets the flag of a message.

```
action:
  name: unflag
```

2.6.7 Read and Unread

The `mark_read` action sets the message as seen or read.

```
action:
  name: mark_read
```

The `mark_unread` action sets the message as unseen or unread.

```
action:
  name: mark_unread
```

2.7 Complete example configuration file

Here's an example of a configuration file with all the possible parts.

```
server:
  hostname: imap.gmail.com
  username: user@example.com
  password: xxxxxxxxxxxxx
```

(continues on next page)

(continued from previous page)

```
trash-mailbox: "[Gmail]/Trash"

mailboxes:
- name: INBOX
  rules:
  - headers:
    - name: "from"
      substring: user1@example.com
    action:
      name: "move"
      dest-mailbox: "User1 correspondence"
  - headers:
    - name: recipient
      substring: dev-team
    - name: subject
      substring: "[Django] ERROR"
    action:
      name: "move"
      dest-mailbox: "Django Errors"
```

CHAPTER 3

Running

Run `imapautofiler` on the command line.

```
$ imapautofiler -h
usage: imapautofiler [-h] [-v] [--debug] [-c CONFIG_FILE] [--list-mailboxes]

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         report more details about what is happening
  --debug               turn on imaplib debugging output
  -c CONFIG_FILE, --config-file CONFIG_FILE
  --list-mailboxes      instead of processing rules, print a list of mailboxes
```

When run with no arguments, it reads the default configuration file and processes the rules.

```
$ imapautofiler
Password for my-user@example.com
Trash: 13767 (Re: spam message from disqus comment) to INBOX.Trash
Move: 13771 (Re: [Openstack-operators] [deployment] [oslo] [ansible] [tripleo]␣
↪[kolla] [helm] Configuration management with etcd / confd) to INBOX.OpenStack.Misc␣
↪Lists
imapautofiler: encountered 10 messages, processed 2
```

Different IMAP servers may use different naming conventions for mailbox hierarchies. Use the `--list-mailboxes` option to the command line program to print a list of all of the mailboxes known to the account.

```
$ imapautofiler --list-mailboxes
Password for my-user@example.com:
INBOX
INBOX.Archive
INBOX.Conferences.PyCon-Organizers
INBOX.ML.TIP
INBOX.ML.python-announce-list
INBOX.OpenStack.Dev List
```

(continues on next page)

(continued from previous page)

```
INBOX.PSF
INBOX.Personal
INBOX.PyATL
INBOX.Sent Items
INBOX.Sent Messages
INBOX.Trash
INBOX.Work
```

CHAPTER 4

Contributing

4.1 The Basics

The source code and bug tracker for imapautofiler are hosted on github.

The source code is released under the Apache 2.0 license. All patches should use the same license.

When reporting a bug, please specify the version of imapautofiler you are using.

4.2 Message handling rules

imapautofiler operation is driven by *rules* and *actions* defined in the *configuration file*.

Each rule is evaluated by a class derived from *Rule* and implemented in `imapautofiler/rules.py`.

A rule must implement the `check()` method to support the interface defined by the abstract base class. The method receives an email.message.Message instance containing the message being processed. `check()` must return `True` if the rule should be applied to the message, or `False` if it should not.

Each new rule must be handled in the *factory()* function so that when the name of the rule is encountered the correct rule class is instantiated and returned.

4.3 Message handling actions

Each action triggered by a rule is evaluated by a class derived from *Action* and implemented in `imapautofiler/actions.py`.

An action must implement the `invoke()` method to support the interface defined by the abstract base class. The method receives an `IMAPClient` instance (from the imapclient package) connected to the IMAP server being scanned, a string message ID, and an email.message.Message instance containing the message being processed. `invoke()` must perform the relevant operation on the message.

Each new action must be handled in the *factory()* function so that when the name of the action is encountered the correct action class is instantiated and returned.

4.4 API Documentation

4.4.1 The imapautofiler.actions Module

class imapautofiler.actions.**Action**(*action_data*, *cfg*)
Bases: object

Base class

NAME = None

invoke(*conn*, *mailbox_name*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (imapautofiler.client.Client) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report(*conn*, *mailbox_name*, *message_id*, *message*)
Log a message explaining what action will be taken.

class imapautofiler.actions.**Delete**(*action_data*, *cfg*)
Bases: *imapautofiler.actions.Action*

Delete the message immediately.

The action is indicated with the name delete.

NAME = 'delete'

invoke(*conn*, *mailbox_name*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (imapautofiler.client.Client) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report(*conn*, *mailbox_name*, *message_id*, *message*)
Log a message explaining what action will be taken.

class imapautofiler.actions.**Flag**(*action_data*, *cfg*)
Bases: *imapautofiler.actions.Action*

Flag the message.

The action is indicated with the name flag.

NAME = 'flag'

invoke(*conn*, *mailbox_name*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (imapautofiler.client.Client) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report(*conn*, *mailbox_name*, *message_id*, *message*)
Log a message explaining what action will be taken.

class imapautofiler.actions.**MarkRead**(*action_data*, *cfg*)
Bases: *imapautofiler.actions.Action*

Mark the message as read

The action is indicated with the name mark_read.

NAME = 'mark_read'

invoke(*conn*, *mailbox_name*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (imapautofiler.client.Client) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report(*conn*, *mailbox_name*, *message_id*, *message*)
Log a message explaining what action will be taken.

class imapautofiler.actions.**MarkUnread**(*action_data*, *cfg*)
Bases: *imapautofiler.actions.Action*

Mark the message as unread

The action is indicated with the name mark_unread.

NAME = 'mark_unread'

invoke(*conn*, *mailbox_name*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (imapautofiler.client.Client) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report(*conn*, *mailbox_name*, *message_id*, *message*)
Log a message explaining what action will be taken.

class imapautofiler.actions.**Move**(*action_data*, *cfg*)
Bases: *imapautofiler.actions.Action*

Move the message to a different folder.

The action is indicated with the name `move`.

The action data must contain a `dest-mailbox` entry with the name of the destination mailbox.

The `dest-mailbox` value can contain jinja2 template directives using the headers of the message. For example:

```
dest-mailbox: "archive.{{ date.year }}"
```

will extract the year value from the date header of the message and insert it into the destination mailbox path.

Header names are always all lower case and - is replaced by _.

NAME = 'move'

invoke(*conn*, *src_mailbox*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (*imapautofiler.client.Client*) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report(*conn*, *src_mailbox*, *message_id*, *message*)
Log a message explaining what action will be taken.

class imapautofiler.actions.**Sort**(*action_data*, *cfg*)
Bases: *imapautofiler.actions.Action*

Move the message based on parsing a destination from a header.

The action is indicated with the name `sort`.

The action may contain a `header` entry to specify the name of the mail header to examine to find the destination. The default is to use the `to` header.

The action data may contain a `dest-mailbox-regex` entry for parsing the header value to obtain the destination mailbox name. If the regex has one match group, that substring will be used. If the regex has more than one match group, the `dest-mailbox-regex-group` option must specify which group to use (0-based numerical index). The default pattern is `([\w-+]+)@` to match the first part of an email address.

The action data must contain a `dest-mailbox-base` entry with the base name of the destination mailbox. The actual mailbox name will be constructed by appending the value extracted via `dest-mailbox-regex` to the `dest-mailbox-base` value. The `dest-mailbox-base` value should contain the mailbox separator character (usually `.`) if the desired mailbox is a sub-folder of the name given.

The `dest-mailbox-base` may include jinja2 template instructions, which are evaluated before the suffix is added to the base. Refer to the description of the `move` action for more details about template evaluation.

NAME = 'sort'

invoke(*conn*, *src_mailbox*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (imapautofiler.client.Client) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report (*conn*, *src_mailbox*, *message_id*, *message*)
Log a message explaining what action will be taken.

class imapautofiler.actions.**SortMailingList** (*action_data*, *cfg*)
Bases: *imapautofiler.actions.Sort*

Move the message based on the mailing list id.

The action is indicated with the name `sort-mailing-list`.

This action is equivalent to the `sort` action with header set to `list-id` and `dest-mailbox-regex` set to `<?([^.]+)\..*>?`.

NAME = 'sort-mailing-list'

class imapautofiler.actions.**Trash** (*action_data*, *cfg*)
Bases: *imapautofiler.actions.Move*

Move the message to the trashcan.

The action is indicated with the name `trash`.

The action expects the global configuration setting `trash-mailbox`.

NAME = 'trash'

class imapautofiler.actions.**Unflag** (*action_data*, *cfg*)
Bases: *imapautofiler.actions.Action*

Remove the flag setting from the message.

The action is indicated with the name `unflag`.

NAME = 'unflag'

invoke (*conn*, *mailbox_name*, *message_id*, *message*)
Run the action on the message.

Parameters

- **conn** (imapautofiler.client.Client) – connection to mail server
- **mailbox_name** (*str*) – name of the mailbox holding the message
- **message_id** (*str*) – ID of the message to process
- **message** (*email.message.Message*) – the message object to process

report (*conn*, *mailbox_name*, *message_id*, *message*)
Log a message explaining what action will be taken.

imapautofiler.actions.**factory** (*action_data*, *cfg*)
Create an Action instance.

Parameters

- **action_data** (*dict*) – portion of configuration describing the action
- **cfg** (*dict*) – full configuration data

Using the action type, instantiate an action object that can process a message.

4.4.2 The imapautofiler.app Module

imapautofiler.app.**list_mailboxes**(*cfg*, *debug*, *conn*)
Print a list of the mailboxes.

Parameters

- **cfg** (*dict*) – full configuration
- **debug** (*bool*) – flag to control debug output
- **conn** (imapautofiler.client.Client) – IMAP server onnection

Used by the `--list-mailboxes` switch.

imapautofiler.app.**main**(*args=None*)

imapautofiler.app.**process_rules**(*cfg*, *debug*, *conn*, *dry_run=False*)
Run the rules from the configuration file.

Parameters

- **cfg** (*dict*) – full configuration
- **debug** (*bool*) – flag to control debug output
- **conn** (imapautofiler.client.Client) – IMAP server onnection

4.4.3 The imapautofiler.client Module

Mail client API.

class imapautofiler.client.**Client**(*cfg*)
Bases: object

close()
Close the connection, flushing any pending changes.

copy_message(*src_mailbox*, *dest_mailbox*, *message_id*, *message*)
Create a copy of the message in the destination mailbox.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **dest_mailbox** (*src*) – name of the destination mailbox
- **message_id** (*str*) – ID of the message to copy
- **message** (*email.message.Message*) – message instance

delete_message(*src_mailbox*, *message_id*, *message*)
Remove the message.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **message** (*email.message.Message*) – message instance

expunge()
Flush any pending changes.

list_mailboxes()
Return a list of mailbox names.

mailbox_iterate(*mailbox_name*)
Iterate over messages from the mailbox.

Produces tuples of (message_id, message).

move_message(*src_mailbox*, *dest_mailbox*, *message_id*, *message*)
Move the message from the source to the destination mailbox.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **dest_mailbox** (*src*) – name of the destination mailbox
- **message_id** (*str*) – ID of the message to copy
- **message** (*email.message.Message*) – message instance

set_flagged(*src_mailbox*, *message_id*, *message*, *is_flagged*)
Manage the "flagged" flag for the message.

If is_flagged is True, ensure the message is flagged. Otherwise, ensure it is not.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **is_flagged** – whether to flag the message

set_read(*src_mailbox*, *message_id*, *message*, *is_read*)
Manage the "read" flag for the message.

If is_read is True, ensure the message is read. Otherwise, ensure it is not.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **is_read** – whether the message should be marked read

class imapautofiler.client.**IMAPClient**(*cfg*)
Bases: *imapautofiler.client.Client*

close()
Close the connection, flushing any pending changes.

copy_message(*src_mailbox*, *dest_mailbox*, *message_id*, *message*)
Create a copy of the message in the destination mailbox.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **dest_mailbox** (*src*) – name of the destination mailbox
- **message_id** (*str*) – ID of the message to copy
- **message** (*email.message.Message*) – message instance

delete_message(*src_mailbox*, *message_id*, *message*)
Remove the message.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **message** (*email.message.Message*) – message instance

expunge()
Flush any pending changes.

list_mailboxes()
Return a list of folder names.

mailbox_iterate(*mailbox_name*)
Iterate over messages from the mailbox.

Produces tuples of (message_id, message).

set_flagged(*src_mailbox*, *message_id*, *message*, *is_flagged*)
Manage the "flagged" flag for the message.

If is_flagged is True, ensure the message is flagged. Otherwise, ensure it is not.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **is_flagged** – whether to flag the message

set_read(*src_mailbox*, *message_id*, *message*, *is_read*)
Manage the "read" flag for the message.

If is_read is True, ensure the message is read. Otherwise, ensure it is not.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **is_read** – whether the message should be marked read

class imapautofiler.client.**MaildirClient**(*cfg*)
Bases: *imapautofiler.client.Client*

close()
Close the connection, flushing any pending changes.

copy_message(*src_mailbox*, *dest_mailbox*, *message_id*, *message*)
Create a copy of the message in the destination mailbox.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **dest_mailbox** (*src*) – name of the destination mailbox
- **message_id** (*str*) – ID of the message to copy
- **message** (*email.message.Message*) – message instance

delete_message(*src_mailbox*, *message_id*, *message*)
Remove the message.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **message** (*email.message.Message*) – message instance

expunge()
Flush any pending changes.

list_mailboxes()
Return a list of mailbox names.

mailbox_iterate(*mailbox_name*)
Iterate over messages from the mailbox.

Produces tuples of (message_id, message).

set_flagged(*src_mailbox*, *message_id*, *message*, *is_flagged*)
Manage the "flagged" flag for the message.

If is_flagged is True, ensure the message is flagged. Otherwise, ensure it is not.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **is_flagged** – whether to flag the message

set_read(*src_mailbox*, *message_id*, *message*, *is_flagged*)
Manage the "read" flag for the message.

If is_read is True, ensure the message is read. Otherwise, ensure it is not.

Parameters

- **src_mailbox** (*str*) – name of the source mailbox
- **message_id** (*str*) – ID of the message to copy
- **is_read** – whether the message should be marked read

imapautofiler.client.**open_connection**(*cfg*)
Open a connection to the mail server.

4.4.4 The imapautofiler.config Module

imapautofiler.config.**get_config**(*filename*)
Return the configuration data.

Parameters **filename** (*str*) – name of configuration file to read

Read `filename` and parse it as a YAML file, then return the results.

imapautofiler.config.**tobool**(*value*)
Convert config option value to boolean.

4.4.5 The imapautofiler.i18n Module

imapautofiler.i18n.**get_header_value**(*msg*, *name*)
Handle header decoding and return a string we examine.

4.4.6 The imapautofiler.lookup Module

imapautofiler.lookup.**make_lookup_table**(*cls*, *attr_name*)

4.4.7 The imapautofiler.rules Module

class imapautofiler.rules.**And**(*rule_data*, *cfg*)
Bases: *imapautofiler.rules.Rule*

True if all of the sub-rules are true.

The rule data must contain a `rules` list with other rules specifications.

Actions on the sub-rules are ignored.

NAME = 'and'

check(*message*)
Test the rule on the message.

Parameters

- **conn** (*imapclient.IMAPClient*) – connection to IMAP server
- **message** (*email.message.Message*) – the message object to process

class imapautofiler.rules.**HeaderExactValue**(*rule_data*, *cfg*)
Bases: imapautofiler.rules._HeaderMatcher

class imapautofiler.rules.**HeaderExists**(*rule_data*, *cfg*)
Bases: *imapautofiler.rules.Rule*

Looks for a message to have a given header.

NAME = 'header-exists'

check(*message*)
Test the rule on the message.

Parameters

- **conn** (*imapclient.IMAPClient*) – connection to IMAP server
- **message** (*email.message.Message*) – the message object to process

class imapautofiler.rules.**HeaderRegex**(*rule_data*, *cfg*)
Bases: imapautofiler.rules._HeaderMatcher

Implements regular expression matching for headers.

class imapautofiler.rules.**HeaderSubString**(*rule_data*, *cfg*)
Bases: imapautofiler.rules._HeaderMatcher

Implements substring matching for headers.

class imapautofiler.rules.**Headers**(*rule_data*, *cfg*)
Bases: *imapautofiler.rules.Rule*

True if all of the headers match.

The rule data must contain a `headers` list of mappings containing a `name` for the header itself and either a `substring` key mapped to a simple string or a `regex` key mapped to a regular expression to be matched against the value of the header.

NAME = 'headers'

check(*message*)
Test the rule on the message.

Parameters

- **conn** (*imapclient.IMAPClient*) – connection to IMAP server
- **message** (*email.message.Message*) – the message object to process

class imapautofiler.rules.**IsMailingList**(*rule_data*, *cfg*)
Bases: *imapautofiler.rules.HeaderExists*

Looks for a message to have a given header.

NAME = 'is-mailing-list'

class imapautofiler.rules.**Or**(*rule_data*, *cfg*)
Bases: *imapautofiler.rules.Rule*

True if any one of the sub-rules is true.

The rule data must contain a `rules` list with other rules specifications.

Actions on the sub-rules are ignored.

NAME = 'or'

check(*message*)
Test the rule on the message.

Parameters

- **conn** (*imapclient.IMAPClient*) – connection to IMAP server
- **message** (*email.message.Message*) – the message object to process

class imapautofiler.rules.**Recipient**(*rule_data*, *cfg*)
Bases: *imapautofiler.rules.Or*

True if any recipient sub-rule matches.

The rule data must contain a `recipient` mapping containing either a `substring` key mapped to a simple string or a `regex` key mapped to a regular expression.

NAME = 'recipient'

class imapautofiler.rules.**Rule**(*rule_data*, *cfg*)
Bases: `object`

Base class

NAME = None

check(*message*)
Test the rule on the message.

Parameters

- **conn** (*imapclient.IMAPClient*) – connection to IMAP server
- **message** (*email.message.Message*) – the message object to process

get_action()

class imapautofiler.rules.**TimeLimit**(*rule_data*, *cfg*)
Bases: *imapautofiler.rules.Rule*

True if message is older than the specified 'age' measured in number of days.

NAME = 'time-limit'

check(*message*)
Test the rule on the message.

Parameters

- **conn** (*imapclient.IMAPClient*) – connection to IMAP server
- **message** (*email.message.Message*) – the message object to process

imapautofiler.rules.**factory**(*rule_data*, *cfg*)
Create a rule processor.

Parameters

- **rule_data** (*dict*) – portion of configuration describing the rule
- **cfg** (*dict*) – full configuration data

Using the rule type, instantiate a rule processor that can check the rule against a message.

4.4.8 The imapautofiler.secrets Module

class imapautofiler.secrets.**AskPassword**(*hostname*, *username*)
Bases: object

get_password()

class imapautofiler.secrets.**FixedPasswordSecret**(*password*)
Bases: object

get_password()

class imapautofiler.secrets.**KeyringPasswordSecret**(*hostname*, *username*)
Bases: object

get_password()

imapautofiler.secrets.**configure_providers**(*cfg*)

imapautofiler.secrets.**get_password**(*cfg*)

4.5 Local Test Maildir

Use `tools/maildir_test_data.py` to create a local test Maildir with a few sample messages. The script requires several dependencies, so for convenience there is a tox environment pre-configured to run it in a virtualenv.

The script requires one argument to indicate the parent directory where the Maildirs should be created.

```
$ tox -e testdata -- /tmp/testdata
```

4.6 Release Notes

This project uses reno for managing release notes. Pull requests with bug fixes and new features should include release notes and documentation updates.

To create a new release note, use the *reno new* command. Use the version of reno that will be used in the automated documentation build by setting up the documentation build locally using *tox*.

```
$ tox -e docs
```

Then, run the reno command from the virtualenv that tox creates, passing a "slug" containing a summary of the fix or feature.

```
$ ./.tox/docs/bin/reno new add-reno
no configuration file in: ./releasenotes/config.yaml, ./reno.yaml
Created new notes file in releasenotes/notes/add-reno-65a040ebe662341a.yaml
```

Finally, edit the file that was created. Fill in the "features" or "fixes" section as appropriate, and remove the rest of the text in the file.

To test the build, you will need to *git add* the new file, then run tox again.

```
$ git add releasenotes/notes/add-reno-65a040ebe662341a.yaml
$ tox -e docs
```

The output will appear in the file *doc/build/html/history.html*.

Note: Refer to the reno documentation for more details about adding, editing, and managing release notes.

CHAPTER 5

History

5.1 1.11.0

5.1.1 New Features

- A configuration option has been added to disable the use of SSL/TLS for servers that do not support it. The default is to always use SSL/TLS. See *Server Connection* for details. Contributed by Samuele Zanon.

5.1.2 Upgrade Notes

- This release drops support for python 3.6 and 3.7.

5.2 1.10.0

5.2.1 New Features

- Add `check_hostname` configuration option to allow connection to sites where the hostname does not match the value in the SSL/TLS certificate. See *Server Connection* for details.
- Add *flag* and *unflag* actions. See *Flag and Unflag* for details.
- Add *mark_read* and *mark_unread* actions. See *Read and Unread* for details.

5.3 1.9.0

5.3.1 New Features

- Start using reno for managing release notes.

5.4 CHANGES

5.4.1 1.12.0

- mergify: initial configuration rules
- client: update ssl handling for newer python 3 and imapclient
- requirements: update imapclient dependency to >=2.2.0 for python 3.10
- packaging: add python 3.10 support to trove classifiers
- tox: use current python version only for default environments
- github: add python 3.10 to CI configuration
- stop building universal wheels

5.4.2 1.11.0

- add a release note for the new option to disable ssl connections
- ignore files created by release notes build
- expand test coverage of rules module
- show coverage report for test modules
- Add configuration to enable/disable ssl
- drop support for python 3.6 and 3.7

5.4.3 1.10.0

- add release note for read/unread actions
- add release note for flag actions
- add mark_read/mark_unread actions
- when –debug is set stop when we see an exception
- fix mailbox client implementation of flagging
- clean up flag/unflag logging
- simplify flag and add unflag action
- Add unit tests for ‘flag’ action
- Add documentation for ‘flag’ action
- Implement ‘flag’ action
- add release note for check_hostname

5.4.4 1.9.0

- add github action for publishing releases
- use default python for pep8 tox target
- remove travis config
- add github workflows for unit tests
- add github workflows for check jobs
- update list of default tox environments
- add pkglint tox target for verifying packaging
- move test commands out of travis.sh to tox.ini
- Add unit tests for config
- Add 'check_hostname' server option
- use the correct default ssl context
- document debian dependencies
- update documentation for templating destination folders
- add templating to the sort action
- remove verbose flag from pytest call
- ensure that if a destination mailbox does not exist we create it
- add jinja2 templates to move action
- add python 3.8 to test matrix
- add separate doc requirement file for rtd build
- add contributing instructions for using reno
- add secrets module to API docs
- configure git depth for travis-ci
- add change history
- fix contributing docs
- move CONTRIBUTING.rst to CONTRIBUTING.md
- remove import to fix pep8 error

5.4.5 1.8.1

- Fix comparison with TZ aware datetime in TimeLimit rule
- update URLs for new location in github org

5.4.6 1.8.0

- add xenial dist for py 3.7 on travis
- have travis script show what is installed

- set minimums for test packages
- use yaml safe loader
- use assertEqual instead of assertEquals
- drop direct use of testtools
- fix warning for strings with unusual escapes
- update trove classifiers
- drop python 3.5 and add 3.7
- perform substring matches without regard to case

5.4.7 1.7.0

- decode message subjects before logging
- switch rule loggers to use NAME
- add –dry-run option
- remove debug print statement
- switch action log messages to use action name directly
- add python 3.6 to the default environment list for tox
- fix factory tests so they don't break when new items are registered

5.4.8 1.6.0

- use a separate attribute for i18n test message in test base class
- ignore .eggs directory
- uninstall nose and mock in travis but leave pytest
- ignore tests in coverage output
- switch from testrepository to pytest
- TimeLimit Rule
- case fix for IMAP and fix lint issues
- Allow more imap configuration via autofiler config

5.4.9 1.5.0

- fix indentation of trash-mailbox setting in example
- link to the keyring documentation
- Add support for using the keyring module to store the IMAP password
- restore the api documentation

5.4.10 1.4.1

- add home-page and description to setup.cfg

5.4.11 1.4.0

- do not check in automatically generated documentation files
- document sort and sort-mailing-list actions
- make header exact match rule to work like other header rules
- add i18n support to sort actions
- extend i18n tests to substring and regex matching rules
- revert logging in header check method
- add internationalized header support
- add a name to the and rule for the lookup table
- implement “and” rule
- automate building the lookup tables for factories

5.4.12 1.3.0

- fix pep8 error
- do not assume a mailbox separator in sort action
- make sort-mailing-list more a general sort action
- add sort-mailing-list action
- add a rule for checking if a message is from a mailing list
- add a rule for checking if a header exists
- Add documentation of mailbox list and example configuration
- do not die if there is an error handling one message
- be explicit about the code block type in config docs

5.4.13 1.2.1

- use universal wheels

5.4.14 1.2.0

- check in the docs generated by pbr
- add tool for creating dummy maildir dataset for testing
- add support for local maildir folders
- create a wrapper class for the server connection
- add/update docstrings for classes

- add basic contributor docs for rules and actions
- move flake8 dependency to extras so it is installed by travis
- have pbr and sphinx automatically generate API docs for classes
- wrap travis with script to support more complex build configurations
- configure travis to test doc build
- add tox environment to test sphinx build
- ignore .coverage output files
- configure travis-ci
- fix docstring for get_message

5.4.15 1.1.1

- add contributing instructions
- add the documentation link to the readme
- use the default docs theme
- add documentation

5.4.16 1.1.0

- prompt the user for a password if none is given

5.4.17 1.0.0

- add 'recipient' rules to cut down on repetition
- report on how many messages were processed at the end of the run
- add regex support for header matching
- add tests for actions
- make Action an abstract base class
- move test message to property of base class
- add tests for Headers
- add tests for HeaderSubString
- use Or directly in tests
- simplify rules tests to decouple Or from HeaderSubstring
- show missing coverage lines in report output
- make Rule an abstract base class
- expand Or rule tests
- add test coverage report
- start writing unit tests

- protect against missing header
- add –list-mailboxes and ‘trash’ action
- abstract out the actions
- start refactoring rules into classes
- support multiple types of actions
- switch to imapclient library, which uses uids
- semi-working version, gets confused after an expunge
- clean up some of the local debug messages
- separate imap debug from local verbose output
- simple rule application
- fix typo in packaging file
- initial structural commit

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

Index

K

L

M

N

O

P

R

S

T

U