isync

isync (http://isync.sourceforge.net/) is a command line application to synchronize mailboxes; it supports Maildir and IMAP4 mailboxes. New messages, message deletions and flag changes can be propagated both ways.

Synchronization is based on unique message identifiers (UIDs), so no identification conflicts can occur (as opposed to some other mail synchronizers). Synchronization state is kept in one local text file per mailbox pair; multiple replicas of a mailbox can be maintained.

Note: isync is the name of the project, mbsync is the name of the executable

Contents

Installing

Install the `isync (https://archlinux.org/packages/?name=isync)` package.

Configuring

Note: Google appears to block isync from downloading emails by default. If you have 2-step authentication enabled, you need to **set up an app password (https://myaccount.google.co**

m/apppasswords) and use that with isync, otherwise you need to go to Google's Security Page (https://myaccount.google.com/security) and toggle "Allow less secure apps" to "on".

Note: Subfolders setting in MaildirStore now seems to be required to be set: iSync Config SubFolders (http://isync.sourceforge.net/mbsync.html) **SubFolders Legacy** worked as previous unset - Oct 2017

First create and customize the main configuration file using this example `~/.mbsyncrc`:

```
~/.mbsyncrc
```

```
IMAPAccount gmail
# Address to connect to
Host imap.gmail.com
User username@gmail.com
Pass ***************
# To store the password in an encrypted file use PassCmd instead of Pass
# PassCmd "gpg2 -q --for-your-eyes-only --no-tty -d ~/.mailpass.gpg"
#
# Use SSL
SSLType IMAPS
# The following line should work. If you get certificate errors, uncomment the two following lines and re
ad the "Troubleshooting" section.
CertificateFile /etc/ssl/certs/ca-certificates.crt
#CertificateFile ~/.cert/imap.gmail.com.pem
#CertificateFile ~/.cert/Equifax_Secure_CA.pem

IMAPStore gmail-remote
Account gmail

MaildirStore gmail-local
SubFolders Verbatim
# The trailing "/" is important
Path ~/.mail/gmail/
Inbox ~/.mail/gmail/Inbox

Channel gmail
Far :gmail-remote:
Near :gmail-local:
# Exclude everything under the internal [Gmail] folder, except the interesting folders
Patterns * ![Gmail]* "[Gmail]/Sent Mail" "[Gmail]/Starred" "[Gmail]/All Mail"
# Or include everything
#Patterns *
# Automatically create missing mailboxes, both locally and on the server
Create Both
# Sync the movement of messages between folders and deletions, add after making sure the sync works
Expunge Both
# Save the synchronization state files in the relevant directory
SyncState *
```

To get rid of the [Gmail]-Stuff (or [Google Mail] as in my case) in each mailbox name, it's possible to use separate Channels for each directory, and later merge them to a group:

```
~/.mbsyncrc
```

```
Channel sync-googlemail-default
Far :gmail-remote:
Near :gmail-local:
# Select some mailboxes to sync
Patterns "INBOX" "arch"

Channel sync-googlemail-sent
Far :gmail-remote:"[Google Mail]/Gesendet"
Near :gmail-local:sent
Create Near

Channel sync-googlemail-trash
Far :gmail-remote:"[Google Mail]/Papierkorb"
Near :gmail-local:trash
Create Near
```

```
# Get all the channels together into a group.
Group googlemail
Channel sync-googlemail-default
Channel sync-googlemail-sent
Channel sync-googlemail-trash
```

As you can see, name-translations are possible this way, as well.

Usage

First make any folders that were specified as Maildirs.

```
$ mkdir -p ~/.mail/gmail
```

Then to retrieve the mail for a specific channel run:

```
$ mbsync gmail
```

or to retrive the mail for all channels:

```
$ mbsync -a
```

Tips and tricks

Using Path and/or Inbox on NTFS partitions

Since ntfs partitions will not accept ; in a filename, you need to change your InfoDelimiter and your FieldDelimiter to something else, you can achieve this by globaly (outside any store or channel configuration) changing the later, like below:

```
~/.mbsyncrc
```

```
FieldDelimiter -
```

Calling mbsync automatically

With a timer

If you want to automatically synchronize your mailboxes, isync can be started automatically with a systemd/User unit. The following service file can start the `mbsync` command:

```
~/.config/systemd/user/mbsync.service
```

```
[Unit]
Description=Mailbox synchronization service

[Service]
```

```
Type=oneshot
ExecStart=/usr/bin/mbsync -Va
```

The following timer configures `mbsync` to be started 2 minutes after boot, and then every 5 minutes:

```
~/.config/systemd/user/mbsync.timer
```

```
[Unit]
Description=Mailbox synchronization timer

[Timer]
OnBootSec=2m
OnUnitActiveSec=5m
Unit=mbsync.service

[Install]
WantedBy=timers.target
```

Once those two files are created, **reload** systemd, then **enable** and **start** `mbsync.timer`, adding the `--user` flag to `systemctl`.

Tip: The mbsync service now only runs after login. It's also possible to launch the systemd-user instances after boot if you configure **Systemd/User#Automatic start-up of systemd user instances**.

Integration with notmuch or mu4e

If you want to run **notmuch** or mu/mu4e after automatically synchronizing your mails, it is preferable to modify the above `mbsync.service` by adding a post-start hook, like below:

```
~/.config/systemd/user/mbsync.service
```

```
[Unit]
Description=Mailbox synchronization service

[Service]
Type=oneshot
ExecStart=/usr/bin/mbsync -Va
ExecStartPost=/usr/bin/notmuch new
```

You can also index `mu` by changing the `ExecStartPost` line to `ExecStartPost=/usr/bin/mu index`, or to `ExecStartPost=/usr/bin/emacsclient -e '(mu4e-update-index)'` if you are running emacsclient and would like to index `mu4e`.

This modification assumes that you have already setup notmuch or mu/mu4e for your user. If the ExecStart command does not execute successfully, the ExecStartPost command will not execute, so be aware of this!

With imapnotify

IMAP IDLE is a way to get **push notifications** to download new email, rather than polling the server intermittently. This has the advantage of saving bandwidth and delivering your mail as soon as it's available. Isync does not have native IDLE suport, but we can use a program like **imapnotify (ht**

tps://www.npmjs.com/package/imapnotify) to call mbsync when you receive new email. For this example we will use the goimapnotify (https://archlinux.org/packages/?name=goimapnotify) package which is reported to work better with frequent network interruptions.

Install goimapnotify (https://archlinux.org/packages/?name=goimapnotify) and create a config file for each mail server you want to poll. Note that the file name format, including the ".conf", is necessary if you want to use the provided systemd service:

```
~/.config/imapnotify/gmail.conf
```

```
{
  "host": "imap.gmail.com",
  "port": 993,
  "tls": true,
  "tlsOptions": {
    "rejectUnauthorized": false
  },
  "username": "username@gmail.com",
  "password": "",
    "passwordCmd": "pass gmail | head -n1",
  "onNewMail": "mbsync gmail",
  "onNewMailPost": "",
  "boxes": [ "INBOX" ]
}
```

(You can view the full configuration options in the project's README (https://gitlab.com/shackra/goimapnotify).)

Start/enable the `goimapnotify@gmail.service` user unit.

Note that IMAP IDLE only triggers when new mail arrives, not when there is undownloaded mail on the server. For example, if you receive 100 emails with your computer powered off, then turn on your computer, imapnotify will still not download new mail until you receive another email. For this reason you may want to run mbsync once when you log in.

Using XOAUTH2

Install an XOAUTH2 SASL plugin, like cyrus-sasl-xoauth2-git (https://aur.archlinux.org/packages/cyrus-sasl-xoauth2-git/)AUR.

Then install oauth2token (https://aur.archlinux.org/packages/oauth2token/)AUR and follow its README (https://pypi.org/project/oauth2token/) to configure the account. It will be responsible for getting the current XOAUTH2 token using the account credentials every time mbsync needs to authenticate.

Finally add `AuthMechs XOAUTH2` and `PassCmd "oauth2get <provider> <account>"`, substituting `<provider>` and `<account>` with the values you used for `oauth2create`, to the `IMAPAccount` section in the `.mbsyncrc`.

Troubleshooting

SSL error

If you get the following error:

```
SSL error connecting imap.gmail.com (108.177.125.109:993): self signed certificate
```

Since google enforce SNI when you use TLS 1.3, ensure to run at least isync v1.3.0 See **https://sourceforge.net/p/isync/isync/merge-requests/2/** for more details

If you get certificate related errors like

```
SSL error connecting pop.mail.com (193.222.111.111:143): error:00000012:lib(0):func(0):reason(18)
```

you may need to retrieve the server's certificates manually in order for mbsync to correctly verify it.

Step #1: Get the certificates

```
$ mkdir ~/.cert
$ openssl s_client -connect some.imap.server:port -showcerts 2>&1 < /dev/null | sed -ne '/-BEGIN CERTIFIC
ATE-/,/-END CERTIFICATE-/p' | sed -ne '1,/-END CERTIFICATE-/p' > ~/.cert/some.imap.server.pem
```

This will create a certificate file called `~/.cert/some.imap.server.pem` (e.g. `~/.cert/imap.gmail.com.pem`). Alternatively one can download **get_certs.sh (https://gist.githubusercontent.com/petRUShka/af96ae25ce8280729b9ea049b929f31d/raw/a79471ce8aee3f6d04049039adf870a53a524f7f/get_certs.sh)** and run it:

```
$ mkdir ~/.cert
$ wget https://gist.githubusercontent.com/petRUShka/af96ae25ce8280729b9ea049b929f31d/raw/a79471ce8aee3f6d
04049039adf870a53a524f7f/get_certs.sh
$ sh get_certs.sh some.imap.server port ~/.cert/
```

If you wish to do this manually, you may enter:

```
$ openssl s_client -connect some.imap.server:port -showcerts
```

and it will display output something like:

```
CONNECTED(00000003)
depth=1 C = US, O = Google Inc, CN = Google Internet Authority
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=imap.gmail.com
   i:/C=US/O=Google Inc/CN=Google Internet Authority
-----BEGIN CERTIFICATE-----
MIIDgDCCAumgAwIBAgIKO3MmiwAAAABopTANBgkqhkiG9w0BAQUFADBGMQswCQYD
VQQGEwJVUzETMBEGA1UEChMKR29vZ2xlIEluYzEiMCAGA1UEAxMZR29vZ2xlIElu
dGVybmV0IEF1dGhvcml0eTAeFw0xMjA5MTIxMTU1NDlaFw0xMzA2MDcxOTQzMjda
```

```
MGgxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQHEw1N
b3VudGFpbiBWaWV3MRMwEQYDVQQKEwpHb29nbGUgSW5jMRcwFQYDVQQDEw5pbWFw
LmdtYWlsLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA2OmU9DjI+DFQ
ThqIN4vL6EqZbzH0ejLKcc+zhxsq9BU5hXohSJ1sS5FUU2vReDKk8fd+ZR3cWtpf
CTYAUSvdnz1ZFjESSzyUBmGRqByhoc0yqdfb61NosA4CDaO+z7DtAgKyecqnAJad
TPYYf9aLk/UgJuc6GseitjzFYonXi6ECAwEAAaOCAVEwggFNMB0GA1UdJQQWMBQG
CCsGAQUFBwMBBggrBgEFBQcDAjAdBgNVHQ4EFgQUFuLyTg2NcsyaEESytZbLbQan
YIowHwYDVR0jBBgwFoAUv8Aw6/VDET5nup6R+/xq2uNrEiQwWwYDVR0fBFQwUjBQ
oE6gTIZKaHR0cDovL3d3dy5nc3RhdGljLmNvbS9Hb29nbGVJbnRlcm5ldEF1dGhv
cml0eS9Hb29nbGVJbnRlcm5ldEF1dGhvcml0eS5jcmwwZgYIKwYBBQUHAQEEWjBY
MFYGCCsGAQUFBzAChkpodHRwOi8vd3d3LmdzdGF0aWMuY29tL0dvb2dsZUludGVy
bmV0QXV0aG9yaXR5L0dvb2dsZUludGVybmV0QXV0aG9yaXR5LmNydDAMBgNVHRMB
Af8EAjAAMBkGA1UdEQQSMBCCDmltYXAuZ21haWwuY29tMA0GCSqGSIb3DQEBBQUA
A4GBAC1LV7tM6pcyVJLcwdPml4DomtowsjTrqvy5ZFa3SMKANK0iZBgFu74O0THX
8SxP/vn4eAs0yRQxcT1ZuoishLGQl5NoimLaQ4BGQnzFQHDJendfaVKDl21GenJp
is72sIrAeprsVU8PbNsllUamWsIjKr3DH5xQdH54hDtzQojY
-----END CERTIFICATE-----
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority
   i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
-----BEGIN CERTIFICATE-----
MIICsDCCAhmgAwIBAgIDC2dxMA0GCSqGSIb3DQEBBQUAME4xCzAJBgNVBAYTAlVT
MRAwDgYDVQQKEwdFcXVpZmF4MS0wKwYDVQQLEyRFcXVpZmF4IFNlY3VyZSBDZXJ0
aWZpY2F0ZSBBdXRob3JpdHkwHhcNMDkwNjA4MjA0MzI3WhcNMTMwNjA3MTk0MzI3
WjBGMQswCQYDVQQGEwJVUzETMBEGA1UEChMKR29vZ2xlIEluYzEiMCAGA1UEAxMZ
R29vZ2xlIEludGVybmV0IEF1dGhvcml0eTCBnzANBgkqhkiG9w0BAQEFAAOBjQAw
gYkCgYEAye23pIucV+eEPkB9hPSP0XFjU5nneXQUr0SZMyCSjXvlKAy6rWxJfoNf
NFlOCnowzdDXxFdF7dWq1nMmzq0yE7jXDx07393cCDaob1FEm8rWIFJztyaHNWrb
qeXUWaUr/GcZOfqTGBhs3t0lig4zFEfC7wFQeeT9adGnwKziV28CAwEAAaOBozCB
oDAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFL/AMOv1QxE+Z7qekfv8atrjaxIk
MB8GA1UdIwQYMBaAFEjmaPkr0rKV10fYIyAQTzOYkJ/UMBIGA1UdEwEB/wQIMAYB
Af8CAQAwOgYDVR0fBDMwMTAvoC2gK4YpaHR0cDovL2NybC5nZW90cnVzdC5jb20v
Y3Jscy9zZWN1cmVjYS5jcmwwDQYJKoZIhvcNAQEFBQADgYEAuIojxkiWsRF8YHde
BZqrocb6ghwYB8TrgbCoZutJqOkM0ymt9e8kTP3kS8p/XmOrmSfLnzYhLLkQYGfN
0rTw8Ktx5YtaiScRhKqOv5nwnQkhClIZmloJ0pC3+gz4fniisIWvXEyZ2VxVKfml
UUIuOss4jHg7y/j7lYe8vJD5UDI=
-----END CERTIFICATE-----
---
Server certificate
subject=/C=US/ST=California/L=Mountain View/O=Google Inc/CN=imap.gmail.com
issuer=/C=US/O=Google Inc/CN=Google Internet Authority
---
No client certificate CA names sent
---
SSL handshake has read 2108 bytes and written 350 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-RC4-SHA
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.1
    Cipher    : ECDHE-RSA-RC4-SHA
    Session-ID: 77136647F42633D82DEDFBB9EB62AB516547A3697D83BD1884726034613C1C09
    Session-ID-ctx:
    Master-Key: 635957FBA0762B10694560488905F73BDD2DB674C41970542ED079446F27234E2CA51CF26938B8CA56DF5BBC7
1E429A7
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 100800 (seconds)
    TLS session ticket:
    0000 - d6 5b a0 a7 10 0e 64 04-72 93 7c 9f 94 fa 07 57   .[....d.r.|....W
    0010 - f1 8b 9d 24 8b 9d 1b f3-a8 b1 4d 2c a9 00 e1 82   ...$......M,....
    0020 - 00 83 1e 3f e5 f2 b2 2c-d2 a8 87 83 16 02 0d 1e   ...?...,........
    0030 - bf b6 c1 d6 75 21 04 e6-63 6b ab 5b ed 94 7a 30   ....u!..ck.[..z0
    0040 - 1a d0 aa 44 c2 04 9b 10-06 28 b5 7b a0 43 a6 0d   ...D.....(.{.C..
    0050 - 3b 4a 85 1f 2e 07 0a e1-32 9b bd 5d 65 41 4c e2   ;J......2..]eAL.
    0060 - 7c d7 43 ec c4 18 77 53-b5 d4 84 b4 c9 bd 51 d6   |.C...wS......Q.
    0070 - 2d 4f 2e 10 a6 ed 38 c5-8e 9d f8 8b 8a 63 3f 7b   -O....8......c?{
    0080 - ee e6 b8 bf 7a f8 b8 e8-47 92 84 f1 9b 0c 63 30   ....z...G.....c0
    0090 - 76 d8 e1 44                                       v..D

    Start Time: 1352632558
    Timeout   : 300 (sec)
    Verify return code: 20 (unable to get local issuer certificate)
---
* OK Gimap ready for requests from 108.78.162.240 o67if11168976yhc.67
```

Simply copy the first block that begins with `-----BEGIN CERTIFICATE-----` and ends with `-----END CERTIFICATE-----`, paste into a file, and save with a .pem extension (this is necessary for the next step). Older instructions state that, with Gmail, both certificate blocks must be saved but on testing this was found to be unnecessary.

Now, copy the root issuer certificate to your local certificate folder. In this example (Gmail), the root issuer is Equifax Secure Certificate Authority. This certificate is included in the **ca-certificates (https://archlinux.org/packages/?name=ca-certificates)** package.

```
cp /usr/share/ca-certificates/mozilla/Equifax_Secure_CA.crt ~/.cert/Equifax_Secure_CA.pem
```

Step #2: Setup mbsync

Configure mbsync to use that certificate:

```
~/.mbsyncrc
```

```
IMAPAccount gmail
Host imap.gmail.com
# ...
CertificateFile ~/.cert/imap.gmail.com.pem
```

BAD Command with Exchange 2003

When connecting to an MS Exchange 2003 server, there could be problems when using pipelining (i.e. executing multiple imap commands concurrently). Such an issue could look as follows:

```
sample output of `mbsync -V exchange'

>>> 9 SELECT "arch"^M
* 250 EXISTS
* 0 RECENT
* FLAGS (\Seen \Answered \Flagged \Deleted \Draft $MDNSent)
* OK [PERMANENTFLAGS (\Seen \Answered \Flagged \Deleted \Draft $MDNSent)] Permanent flags
* OK [UNSEEN 241] Is the first unseen message
* OK [UIDVALIDITY 4352] UIDVALIDITY value
9 OK [READ-WRITE] SELECT completed.
>>> 10 UID FETCH 1:1000000000 (UID FLAGS)^M
* 1 FETCH (UID 1 FLAGS (\Seen \Answered))
* 2 FETCH (UID 2 FLAGS (\Seen \Answered))
...
* 249 FETCH (UID 696 FLAGS ())
* 250 FETCH (UID 697 FLAGS (\Seen))
10 OK FETCH completed.
>>> 11 APPEND "arch" (\Seen) {4878+}^M
(1 in progress) >>> 12 UID FETCH 697 (BODY.PEEK[])^M
(2 in progress) >>> 13 UID STORE 696 +FLAGS.SILENT (\Deleted)^M
12 BAD Command is not valid in this state.
```

So command 9 is to select a new folder, command 10 checks the mail and commands 11, 12 and 13 run in parallel, writing/getting/flagging a mail. In this case, the Exchange server would terminate the connection after the BAD return value and go on to the next channel. (And if all went well in this channel, mbsync would return with 0.) After setting

```
PipelineDepth 1
```

in the IMAPStore config part of the Exchange, this problem did not occur any more.

Emails on remote server have the wrong date

This fix works when syncing with fastmail, but it likely applies to other services as well.

If you move an email to a new folder using an email client, and mbsync causes the email to appear with the wrong date on the server, add this to your configuration file:

```
 CopyArrivalDate yes
```

For example, without this setting, moving an old email from Inbox to Archive using mu4e and then syncing to fastmail with mbsync will cause the email to appear in Archive but with the date of the sync.

mbsync uses mtime of email message when uploading from maildir to imap server. You can use fix_maildir_mail_mtime.py (https://gist.github.com/artizirk/877ce9d30159323aac037e2a2af74509) script to set mtime from email header.

External links

- Home page (http://isync.sourceforge.net/)
- Sourceforge page (https://sourceforge.net/projects/isync/)
- backing up gmail with mbsync (https://kevin.deldycke.com/2012/08/gmail-backup-mbsync/)
- How To Verify SSL Certificate From A Shell Prompt (https://www.cyberciti.biz/faq/test-ssl-certificates-diagnosis-ssl-certificate/)

Retrieved from "https://wiki.archlinux.org/index.php?title=Isync&oldid=710652"