NAME

OfflineIMAP – Powerful IMAP/Maildir synchronization and reader support

SYNOPSIS

offlineimap [**-1**] [**-P** *profiledir*] [**-a** *accountlist*] [**-c** *configfile*] [**-d** *debugtype[,...]*] [**-l** *filename*] [**-o**] [**-u** *interface*]

offlineimap -h | --help

DESCRIPTION

OfflineIMAP is a tool to simplify your e-mail reading. With **OfflineIMAP**, you can read the same mailbox from multiple computers. You get a current copy of your messages on each computer, and changes you make one place will be visible on all other systems. For instance, you can delete a message on your home computer, and it will appear deleted on your work computer as well. **OfflineIMAP** is also useful if you want to use a mail reader that does not have IMAP support, has poor IMAP support, or does not provide disconnected operation.

OfflineIMAP is **FAST**; it synchronizes my two accounts with over 50 folders in 3 seconds. Other similar tools might take over a minute, and achieve a less-reliable result. Some mail readers can take over 10 minutes to do the same thing, and some don't even support it at all. Unlike other mail tools, **OfflineIMAP** features a multi-threaded synchronization algorithm that can dramatically speed up performance in many situations by synchronizing several different things simultaneously.

OfflineIMAP is **FLEXIBLE**; you can customize which folders are synced via regular expressions, lists, or Python expressions; a versatile and comprehensive configuration file is used to control behavior; two user interfaces are built-in; fine-tuning of synchronization performance is possible; internal or external automation is supported; SSL and PREAUTH tunnels are both supported; offline (or "unplugged") reading is supported; and esoteric IMAP features are supported to ensure compatibility with the widest variety of IMAP servers.

OfflineIMAP is **SAFE**; it uses an algorithm designed to prevent mail loss at all costs. Because of the design of this algorithm, even programming errors should not result in loss of mail. I am so confident in the algorithm that I use my own personal and work accounts for testing of **OfflineIMAP** pre-release, development, and beta releases. Of course, legally speaking, **OfflineIMAP** comes with no warranty, so I am not responsible if this turns out to be wrong.

METHOD OF OPERATION

OfflineIMAP traditionally operates by maintaining a hierarchy of mail folders in Maildir format locally. Your own mail reader will read mail from this tree, and need never know that the mail comes from IMAP. **OfflineIMAP** will detect changes to the mail folders on your IMAP server and your own computer and bidirectionally synchronize them, copying, marking, and deleting messages as necessary.

With **OfflineIMAP** 4.0, a powerful new ability has been introduced -- the program can now synchronize two IMAP servers with each other, with no need to have a Maildir layer in-between. Many people use this if they use a mail reader on their local machine that does not support Maildirs. People may install an IMAP server on their local machine, and point both **OfflineIMAP** and their mail reader of choice at it. This is often preferable to the mail reader's own IMAP support since **OfflineIMAP** supports many features (offline reading, for one) that most IMAP-aware readers don't. However, this feature is not as time-tested as traditional syncing, so my advice is to stick with normal methods of operation for the time being.

QUICK START

If you have already installed **OfflineIMAP** system-wide, or your system administrator has done that for you, your task for setting up **OfflineIMAP** for the first time is quite simple. You just need to set up your configuration file, make your folder directory, and run it!

You can quickly set up your configuration file. The distribution includes a file *offlineimap.conf.minimal* (Debian users may find this at */usr/share/doc/offlineimap/examples/offlineimap.conf.minimal*) that is a basic example of setting of **OfflineIMAP**. You can simply copy this file into your home directory and name it

.offlineimaprc (note the leading period). A command such as **cp offlineimap.conf.minimal ~/.offlineimaprc** will do it. Or, if you prefer, you can just copy this text to *~/.offlineimaprc*:

```
[general]
accounts = Test

[Account Test]
localrepository = Local
remoterepository = Remote

[Repository Local]
type = Maildir
localfolders = ~/Test

[Repository Remote]
type = IMAP
remotehost = examplehost
remoteuser = jgoerzen
```

Now, edit the *~/.offlineimaprc* file with your favorite editor. All you have to do is specify a directory for your folders to be in (on the *localfolders* line), the host name of your IMAP server (on the *remotehost* line), and your login name on the remote (on the *remoteuser* line). That's it!

To run **OfflineIMAP**, you just have to say **offlineimap** -- it will fire up, ask you for a login password if necessary, synchronize your folders, and exit. See? You can just throw away the rest of this finely-crafted, perfectly-honed manual! Of course, if you want to see how you can make **OfflineIMAP** FIVE TIMES FASTER FOR JUST $19.95 (err, well, $0), you have to read on!

INSTALLATION

If you are reading this document via the "man" command, it is likely that you have no installation tasks to perform; your system administrator has already installed it. If you need to install it yourself, you have three options: a system-wide installation with Debian, system-wide installation with other systems, and a single-user installation. You can download the latest version of **OfflineIMAP** from the **OfflineIMAP** website <URL:http://quux.org/devel/offlineimap/>.

PREREQUISITES

In order to use **OfflineIMAP**, you need to have these conditions satisfied:

- Your mail server must support IMAP. Most Internet Service Providers and corporate networks do, and most operating systems have an IMAP implementation readily available.
- You must have Python version 2.2.1 or above installed. If you are running on Debian GNU/Linux, this requirement will automatically be taken care of for you. If you do not have Python already, check with your system administrator or operating system vendor; or, download it from the Python website <URL:http://www.python.org/>. If you intend to use the Tk interface, you must have Tkinter (python-tk) installed. If you intend to use the SSL interface, your Python must have been built with SSL support.
- Have a mail reader that supports the Maildir mailbox format. Most modern mail readers have this support built-in, so you can choose from a wide variety of mail servers. This format is also known as the "qmail" format, so any mail reader compatible with it will work with **OfflineIMAP**. If you do not have a mail reader that supports Maildir, you can often install a local IMAP server and point both **OfflineIMAP** and your mail reader at it.

SYSTEM-WIDE INSTALLATION, DEBIAN

If you are tracking Debian unstable, you may install **OfflineIMAP** by simply running the following command as root:

apt-get install offlineimap

If you are not tracking Debian unstable, download the Debian .deb package from the **OfflineIMAP** website <URL:http://quux.org/devel/offlineimap/> and then run **dpkg -i** to install the downloaded package. Then, skip to [XRef to CONFIGURATION] below. You will type **offlineimap** to invoke the program.

SYSTEM-WIDE INSTALLATION, OTHER

Download the tar.gz version of the package from the website <URL:http://quux.org/devel/offlineimap/>. Then run these commands, making sure that you are the "root" user first:

tar -zxvf offlineimap_x.y.z.tar.gz
cd offlineimap-x.y.z
python2.2 setup.py install

On some systems, you will need to use **python** instead of **python2.2**. Next, proceed to [XRef to CONFIGURATION] below. You will type **offlineimap** to invoke the program.

SINGLE-ACCOUNT INSTALLATION

Download the tar.gz version of the package from the website <URL:http://quux.org/devel/offlineimap/>. Then run these commands:

tar -zxvf offlineimap_x.y.z.tar.gz
cd offlineimap-x.y.z

When you want to run **OfflineIMAP**, you will issue the **cd** command as above and then type **./offlineimap.py**; there is no installation step necessary.

CONFIGURATION

OfflineIMAP is regulated by a configuration file that is normally stored in *~/.offlineimaprc*. **OfflineIMAP** ships with a file named *offlineimap.conf* that you should copy to that location and then edit. This file is vital to proper operation of the system; it sets everything you need to run **OfflineIMAP**. Full documentation for the configuration file is included within the sample file.

OfflineIMAP also ships a file named *offlineimap.conf.minimal* that you can also try. It's useful if you want to get started with the most basic feature set, and you can read about other features later with *offlineimap.conf*.

OPTIONS

Most configuration is done via the configuration file. Nevertheless, there are a few command-line options that you may set for **OfflineIMAP**.

-1 Disable most multithreading operations and use solely a single-connection sync. This effectively sets the *maxsyncaccounts* and all *maxconnections* configuration file variables to 1.

-P *profiledir*
Sets **OfflineIMAP** into profile mode. The program will create *profiledir* (it must not already exist). As it runs, Python profiling information about each thread is logged into profiledir. Please note: This option is present for debugging and optimization only, and should NOT be used unless you have a specific reason to do so. It will significantly slow program performance, may reduce reliability, and can generate huge amounts of data. You must use the **-1** option when you use **-P**.

-a *accountlist*
Overrides the *accounts* option in the *general* section of the configuration file. You might use this to exclude certain accounts, or to sync some accounts that you normally prefer not to. Separate the accounts by commas, and use no embedded spaces.

-c *configfile*
Specifies a configuration file to use in lieu of the default, *~/.offlineimaprc*.

-d *debugtype[,...]*
Enables debugging for OfflineIMAP. This is useful if you are trying to track down a malfunction or figure out what is going on under the hood. I suggest that you use this with **-1** to make the results more sensible.

-d requires one or more debugtypes, separated by commas. These define what exactly will be debugged, and include three options: *imap*, *maildir*, and *thread*. The *imap* option will enable IMAP protocol stream and parsing debugging. Note that the output may contain passwords, so take care to remove that from the debugging output before sending it to anyone else. The *maildir* option will enable debugging for certain Maildir operations. And *thread* will debug the threading model.

-l *filename*
Enables logging to filename. This will log everything that goes to the screen to the specified file. Additionally, if any debugging is specified with -d, then debug messages will not go to the screen, but instead to the logfile only.

-o Run only once, ignoring all *autorefresh* settings in the configuration file.

-h

--help Show summary of options.

-u *interface*
Specifies an alternative user interface module to use. This overrides the default specified in the configuration file. The pre-defined options are listed in the User Interfaces section.

USER INTERFACES

OfflineIMAP has a pluggable user interface system that lets you choose how the program communicates information to you. There are two graphical interfaces, two terminal interfaces, and two noninteractive interfaces suitable for scripting or logging purposes. The *ui* option in the configuration file specifies user interface preferences. The **-u** command-line option can override the configuration file setting. The available values for the configuration file or command-line are described in this section.

TK.BLINKENLIGHTS

Tk.Blinkenlights is an interface designed to be sleek, fun to watch, and informative of the overall picture of what **OfflineIMAP** is doing. I consider it to be the best general-purpose interface in **OfflineIMAP**.

Tk.Blinkenlights contains, by default, a small window with a row of LEDs, a small log, and a row of command buttons. The total size of the window is very small, so it uses little desktop space, yet it is quite functional. The optional, toggleable, log shows more detail about what is happening and is color-coded to match the color of the lights.

Tk.Blinkenlights is the only user interface that has configurable parameters; see the example *offlineimap.conf* for more details.

Each light in the Blinkenlights interface represents a thread of execution -- that is, a particular task that **OfflineIMAP** is performing right now. The colors indicate what task the particular thread is performing, and are as follows:

Black indicates that this light's thread has terminated; it will light up again later when new threads start up. So, black indicates no activity.

Red (Meaning 1)
is the color of the main program's thread, which basically does nothing but monitor the others. It might remind you of HAL 9000 in 2001.

Gray indicates that the thread is establishing a new connection to the IMAP server.

Purple is the color of an account synchronization thread that is monitoring the progress of the folders in that account (not generating any I/O).

Cyan indicates that the thread is syncing a folder.

Green means that a folder's message list is being loaded.

Blue is the color of a message synchronization controller thread.

Orange
indicates that an actual message is being copied. (We use fuchsia for fake messages.)

Red (meaning 2)
indicates that a message is being deleted.

Yellow / bright orange
indicates that message flags are being added.

Pink / bright red
indicates that message flags are being removed.

Red / Black Flashing
corresponds to the countdown timer that runs between synchronizations.

The name of this interfaces derives from a bit of computer history. Eric Raymond's Jargon File defines *blinkenlights*, in part, as:

> Front-panel diagnostic lights on a computer, esp. a dinosaur. Now that dinosaurs are rare, this term usually refers to status lights on a modem, network hub, or the like.
>
> This term derives from the last word of the famous blackletter-Gothic sign in mangled pseudo-German that once graced about half the computer rooms in the English-speaking world. One version ran in its entirety as follows:
>
> **ACHTUNG! ALLES LOOKENSPEEPERS!**
>
> Das computermachine ist nicht fuer gefingerpoken und mittengrabben. Ist easy schnappen der springenwerk, blowenfusen und poppencorken mit spitzensparken. Ist nicht fuer gewerken bei das dumpkopfen. Das rubbernecken sichtseeren keepen das cotten-pickenen hans in das pockets muss; relaxen und watchen das blinkenlichten.

CURSES.BLINKENLIGHTS
Curses.Blinkenlights is an interface very similar to Tk.Blinkenlights, but is designed to be run in a console window (an xterm, Linux virtual terminal, etc.) Since it doesn't have access to graphics, it isn't quite as pretty, but it still gets the job done.

Please see the Tk.Blinkenlights section above for more information about the colors used in this interface.

TK.VERBOSEUI
Tk.VerboseUI (formerly known as Tk.TkUI) is a graphical interface that presents a variable-sized window. In the window, each currently-executing thread has a section where its name and current status are displayed. This interface is best suited to people running on slower connections, as you get a lot of detail, but for fast connections, the detail may go by too quickly to be useful. People with fast connections may wish to use Tk.Blinkenlights instead.

TTY.TTYUI
TTY.TTYUI interface is for people running in basic, non-color terminals. It prints out basic status messages and is generally friendly to use on a console or xterm.

NONINTERACTIVE.BASIC
Noninteractive.Basic is designed for situations in which **OfflineIMAP** will be run non-attended and the status of its execution will be logged. You might use it, for instance, to have the system run automatically and e-mail you the results of the synchronization. This user interface is not capable of reading a password from the keyboard; account passwords must be specified using one of the configuration file options.

NONINTERACTIVE.QUIET
Noninteractive.Quiet is designed for non-attended running in situations where normal status messages are not desired. It will output nothing except errors and serious warnings. Like Noninteractive.Basic, this user interface is not capable of reading a password from the keyboard; account passwords must be specified using one of the configuration file options.

EXAMPLES

Here are some example configurations for various situations. Please e-mail any other examples you have that may be useful to me.

MULTIPLE ACCOUNTS WITH MUTT

This example shows you how to set up **OfflineIMAP** to synchronize multiple accounts with the mutt mail reader.

Start by creating a directory to hold your folders by running **mkdir ~/Mail**. Then, in your *~/.offlineimaprc*, specify:

accounts = Personal, Work

Make sure that you have both an *[Account Personal]* and an *[Account Work]* section. The local repository for each account must have different *localfolder* path names. Also, make sure to enable *[mbnames]*.

In each local repository section, write something like this:

localfolders = ~/Mail/Personal

Finally, add these lines to your *~/.muttrc*:

```
source ~/path-to-mbnames-muttrc-mailboxes
folder-hook Personal set from="youremail@personal.com"
folder-hook Work set from="youremail@work.com"
set mbox_type=Maildir
set folder=$HOME/Mail
spoolfile=+Personal/INBOX
```

That's it!

UW-IMAPD AND REFERENCES

Some users with a UW-IMAPD server need to use **OfflineIMAP**'s "reference" feature to get at their mailboxes, specifying a reference of "~/Mail" or "#mh/" depending on the configuration. The below configuration from (originally from docwhat@gerf.org) shows using a *reference* of Mail, a *nametrans* that strips the leading Mail/ off incoming folder names, and a *folderfilter* that limits the folders synced to just three.

```
[Account Gerf]
localrepository = GerfLocal
remoterepository = GerfRemote

[Repository GerfLocal]
type = Maildir
localfolders = ~/Mail

[Repository GerfRemote]
type = IMAP
remotehost = gerf.org
ssl = yes
remoteuser = docwhat
reference = Mail
# Trims off the preceeding Mail on all the folder names.
nametrans = lambda foldername: \
        re.sub('^Mail/', '', foldername)
# Yeah, you have to mention the Mail dir, even though it
# would seem intuitive that reference would trim it.
folderfilter = lambda foldername: foldername in [
    'Mail/INBOX',
    'Mail/list/zaurus-general',
```

```
    'Mail/list/zaurus-dev',
    ]
maxconnections = 1
holdconnectionopen = no
```

PYTHONFILE CONFIGURATION FILE OPTION

You can have **OfflineIMAP** load up a Python file before evaluating the configuration file options that are Python expressions. This example is based on one supplied by Tommi Virtanen for this feature.

In *~/.offlineimap.rc*, he adds these options:

```
[general]
pythonfile=~/.offlineimap.py
[Repository foo]
foldersort=mycmp
```

Then, the *~/.offlineimap.py* file will contain:

```
prioritized = ['INBOX', 'personal', 'announce', 'list']

def mycmp(x, y):
  for prefix in prioritized:
    xsw = x.startswith(prefix)
    ysw = y.startswith(prefix)
    if xsw and ysw:
      return cmp(x, y)
    elif xsw:
      return -1
    elif ysw:
      return +1
  return cmp(x, y)

def test_mycmp():
  import os, os.path
  folders=os.listdir(os.path.expanduser('~/data/mail/tv@hq.yok.utu.fi'))
  folders.sort(mycmp)
  print folders
```

This code snippet illustrates how the *foldersort* option can be customized with a Python function from the *pythonfile* to always synchronize certain folders first.

ERRORS

If you get one of some frequently-encountered or confusing errors, please check this section.

UID VALIDITY PROBLEM FOR FOLDER

IMAP servers use a unique ID (UID) to refer to a specific message. This number is guaranteed to be unique to a particular message **forever**. No other message in the same folder will ever get the same UID. UIDs are an integral part of **OfflineIMAP**'s synchronization scheme; they are used to match up messages on your computer to messages on the server.

Sometimes, the UIDs on the server might get reset. Usually this will happen if you delete and then recreate a folder. When you create a folder, the server will often start the UID back from 1. But **OfflineIMAP** might still have the UIDs from the previous folder by the same name stored. **OfflineIMAP** will detect this condition and skip the folder. This is GOOD, because it prevents data loss.

You can fix it by removing your local folder and cache data. For instance, if your folders are under *~/Folders* and the folder with the problem is INBOX, you'd type this:

```
rm -r ~/Folders/INBOX
```

rm -r ~/.offlineimap/Account-*AccountName*
rm -r ~/.offlineimap/Repository-*RepositoryName*

(Of course, replace AccountName and RepositoryName with the names as specified in *~/.offlineimaprc*).

Next time you run **OfflineIMAP**, it will re-download the folder with the new UIDs. Note that the procedure specified above will lose any local changes made to the folder.

Some IMAP servers are broken and do not support UIDs properly. If you continue to get this error for all your folders even after performing the above procedure, it is likely that your IMAP server falls into this category. **OfflineIMAP** is incompatible with such servers. Using **OfflineIMAP** with them will not destroy any mail, but at the same time, it will not actually synchronize it either. (**OfflineIMAP** will detect this condition and abort prior to synchronization.)

This question comes up frequently on the **OfflineIMAP** mailing list <URL:http://lists.complete.org/offlineimap@complete.org/>. You can find a detailed discussion <URL:http://lists.complete.org/offlineimap@complete.org/2003/04/msg00012.html.gz> of the problem there.

OTHER FREQUENTLY ASKED QUESTIONS

There are some other FAQs that might not fit into another section of the document, so they are discussed here.

What platforms does OfflineIMAP run on?

It should run on most platforms supported by Python, which are quite a few.

I'm using Mutt. Other IMAP sync programs require me to use "set maildir_trash=yes". Do I need to do that with OfflineIMAP?

No. **OfflineIMAP** is smart enough to figure out message deletion without this extra crutch. You'll get the best results if you don't use this setting, in fact.

I've upgraded and now OfflineIMAP crashes when I start it up! Why?

You need to upgrade your configuration file. See [XRef to UPGRADING.4.0] at the end of this manual.

How do I specify the names of my folders?

You do not need to. **OfflineIMAP** is smart enough to automatically figure out what folders are present on the IMAP server and synchronize them. You can use the *folderfilter* and *foldertrans* configuration file options to request certain folders and rename them as they come in if you like.

How can I prevent certain folders from being synced?

Use the *folderfilter* option in the configuration file.

How can I add or delete a folder?

OfflineIMAP does not currently provide this feature, but if you create a new folder on the IMAP server, it will be created locally automatically.

Are there any other warnings that I should be aware of?

Yes; see the Notes section below.

What is the mailbox name recorder (mbnames) for?

Some mail readers, such as Mutt, are not capable of automatically determining the names of your mailboxes. **OfflineIMAP** can help these programs by writing the names of the folders in a format you specify. See the example *offlineimap.conf* for details.

Can I synchronize multiple accounts with OfflineIMAP?

Sure. Just name them all in the *accounts* line in the *general* section of the configuration file, and add a per-account section for each one.

Does OfflineIMAP support POP?

No. POP is not robust enough to do a completely reliable multi-machine synchronization like **OfflineIMAP** can do. **OfflineIMAP** will not support it.

Does OfflineIMAP support mailbox formats other than Maildir?

Not at present. There is no technical reason not to; just no demand yet. Maildir is a superior format anyway. However, **OfflineIMAP** can sync between two IMAP servers, and some IMAP servers support other formats. You could install an IMAP server on your local machine and have **OfflineIMAP** sync to that.

[technical] Why are your Maildir message filenames so huge?

OfflineIMAP has two relevant principles: 1) never modifying your messages in any way and 2) ensuring 100% reliable synchronizations. In order to do a reliable sync, **OfflineIMAP** must have a way to uniquely identify each e-mail. Three pieces of information are required to do this: your account name, the folder name, and the message UID. The account name can be calculated from the path in which your messages are. The folder name can usually be as well, BUT some mail clients move messages between folders by simply moving the file, leaving the name intact.

So, **OfflineIMAP** must store both a UID folder ID. The folder ID is necessary so **OfflineIMAP** can detect a message moved to a different folder. **OfflineIMAP** stores the UID (U= number) and an md5sum of the foldername (FMD5= number) to facilitate this.

What is the speed of OfflineIMAP's sync?

OfflineIMAP versions 2.0 and above contain a multithreaded system. A good way to experiment is by setting *maxsyncaccounts* to 3 and *maxconnections* to 3 in each account clause.

This lets OfflineIMAP open up multiple connections simultaneously. That will let it process multiple folders and messages at once. In most cases, this will increase performance of the sync.

Don't set the number too high. If you do that, things might actually slow down as your link gets saturated. Also, too many connections can cause mail servers to have excessive load. Administrators might take unkindly to this, and the server might bog down. There are many variables in the optimal setting; experimentation may help.

An informal benchmark yields these results for my setup:

- 10 minutes with MacOS X Mail.app "manual cache"
- 5 minutes with GNUS agent sync
- 20 seconds with OfflineIMAP 1.x
- 9 seconds with OfflineIMAP 2.x
- 3 seconds with OfflineIMAP 3.x "cold start"
- 2 seconds with OfflineIMAP 3.x "held connection"

CONFORMING TO

- Internet Message Access Protocol version 4rev1 (IMAP 4rev1) as specified in RFC2060 and RFC3501
- CRAM-MD5 as specified in RFC2195
- Maildir as specified in the Maildir manpage <URL:http://www.qmail.org/qmail-manual-html/man5/maildir.html> and the qmail website <URL:http://cr.yp.to/proto/maildir.html>.
- Standard Python 2.2.1 as implemented on POSIX-compliant systems.

NOTES

DELETING LOCAL FOLDERS

OfflineIMAP does a two-way synchronization. That is, if you make a change to the mail on the server, it will be propagated to your local copy, and vise-versa. Some people might think that it would be wise to just delete all their local mail folders periodically. If you do this with **OfflineIMAP**, remember to also remove your local status cache (*~/.offlineimap* by default). Otherwise, **OfflineIMAP** will take this as an intentional deletion of many messages and will interpret your action as requesting them to be deleted from the server as well. (If you don't understand this, don't worry; you probably won't encounter this situation)

MULTIPLE INSTANCES

OfflineIMAP is not designed to have several instances (for instance, a cron job and an interactive invocation) run over the same mailbox simultaneously. It will perform a check on startup and abort if another **OfflineIMAP** is already running. If you need to schedule synchronizations, please use the *autorefresh* settings rather than cron. Alternatively, you can set a separate *metadata* directory for each instance.

COPYING MESSAGES BETWEEN FOLDERS

Normally, when you copy a message between folders or add a new message to a folder locally, **OfflineIMAP** will just do the right thing. However, sometimes this can be tricky -- if your IMAP server does not provide the SEARCH command, or does not return something useful, **OfflineIMAP** cannot determine the new UID of the message. So, in these rare instances, OfflineIMAP will upload the message to the IMAP server and delete it from your local folder. Then, on your next sync, the message will be re-downloaded with the proper UID. **OfflineIMAP** makes sure that the message was properly uploaded before deleting it, so there should be no risk of data loss.

USE WITH EVOLUTION

OfflineIMAP can work with Evolution. To do so, first configure your **OfflineIMAP** account to have **sep = /** in its configuration. Then, configure Evolution with the "Maildir-format mail directories" server type. For the path, you will need to specify the name of the top-level folder **inside** your **OfflineIMAP** storage location. You're now set!

USE WITH KMAIL

At this time, I believe that **OfflineIMAP** with Maildirs is not compatible with KMail. KMail cannot work in any mode other than to move all messages out of all folders immediately, which (besides being annoying and fundamentally broken) is incompatible with **OfflineIMAP**.

However, I have made KMail version 3 work well with **OfflineIMAP** by installing an IMAP server on my local machine, having **OfflineIMAP** sync to that, and pointing KMail at the same server.

MAILING LIST

There is an OfflineIMAP mailing list available. To subscribe, send the text "Subscribe" in the subject of a mail to offlineimap-request@complete.org. To post, send the message to offlineimap@complete.org. Archives are available at
<URL:http://lists.complete.org/offlineimap@complete.org/>.

BUGS

Reports of bugs should be sent via e-mail to the **OfflineIMAP** bug-tracking system (BTS) at offlineimap@bugs.complete.org or submitted online using the web interface <URL:http://bugs.complete.org/>.

The Web site also lists all current bugs, where you can check their status or contribute to fixing them.

UPGRADING TO 4.0

If you are upgrading from a version of **OfflineIMAP** prior to 3.99.12, you will find that you will get errors when **OfflineIMAP** starts up (relating to ConfigParser or AccountHashGenerator) and the configuration file. This is because the config file format had to change to accommodate new features in 4.0. Fortunately, it's not difficult to adjust it to suit.

First thing you need to do is stop any running **OfflineIMAP** instance, making sure first that it's synced all your mail. Then, modify your *~/.offlineimaprc* file. You'll need to split up each account section (make sure that it now starts with "Account ") into two Repository sections (one for the local side and another for the remote side.) See the files *offlineimap.conf.minimal* and *offlineimap.conf* in the distribution if you need more assistance.

OfflineIMAP's status directory area has also changed. Therefore, you should delete everything in ~/.offlineimap as well as your local mail folders.

When you start up **OfflineIMAP** 4.0, it will re-download all your mail from the server and then you can continue using it like normal.

COPYRIGHT

OfflineIMAP, and this manual, are Copyright (C) 2002, 2003 John Goerzen.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

imaplib.py comes from the Python dev tree and is licensed under the GPL-compatible PSF license as stated in the file *COPYRIGHT* in the **OfflineIMAP** distribution.

AUTHOR

OfflineIMAP, its libraries, documentation, and all included files, except where noted, was written by John Goerzen <jgoerzen@complete.org> and copyright is held as stated in the COPYRIGHT section.

OfflineIMAP may be downloaded, and information found, from its homepage via either Gopher <URL:gopher://quux.org/1/devel/offlineimap> or HTTP <URL:http://quux.org/devel/offlineimap>.

OfflineIMAP may also be downloaded using Subversion. Additionally, the distributed tar.gz may be updated with a simple "svn update" command; it is ready to go. For information on getting OfflineIMAP with Subversion, please visit the complete.org Subversion page <URL:http://svn.complete.org/>.

SEE ALSO

mutt(1), **python**(1)

HISTORY

Detailed history may be found in the file ChangeLog in the **OfflineIMAP** distribution. Feature and bug histories may be found in the file debian/changelog which, despite its name, is not really Debian-specific. This section provides a large overview.

Development on **OfflineIMAP** began on June 18, 2002. Version 1.0.0 was released three days later on June 21, 2002. Point releases followed, including speed optimizations and some compatibility fixes.

Version 2.0.0 was released on July 3, 2002, and represented the first time the synchronization became multithreaded and, to the best of my knowledge, the first multithreaded IMAP syncrhonizing application in existance. The last 2.0.x release, 2.0.8, was made on July 9.

Version 3.0.0 was released on July 11, 2002, and introduced modular user interfaces and the first GUI interface for **OfflineIMAP**. This manual also was introduced with 3.0.0, along with many command-line options. Version 3.1.0 was released on July 21, adding the Noninteractive user interfaces, profiling support, and several bugfixes. 3.2.0 was released on July 24, adding support for the Blinkenlights GUI interface. **OfflineIMAP** entered maintenance mode for awhile, as it had reached a feature-complete milestone in my mind.

The 3.99.x branch began in on October 7, 2002, to begin work for 4.0. The Curses.Blinkenlights interface was added in 3.99.6, and many architectural changes were made.

4.0.0 was released on July 18, 2003, including the ability to synchronize directly between two IMAP servers, the first re-architecting of the configuration file to refine the notion of an account, and the new Curses interface.