feed2exec Documentation

Release 0.14.1.dev1+gbad1d89

Antoine Beaupré

Mar 21, 2019

Contents

`feed2exec` is a simple program that runs custom actions on new RSS feed items (or whatever feedparser can read). It currently has support for writing into mailboxes (Maildir folders) or executing commands, but more actions can be easily implemented through plugins. Email are saved as multipart plain/HTML and can be sent to arbitrary folders.

CHAPTER 1

Examples

Simple run with no side effects:

```
feed2exec parse https://www.nasa.gov/rss/dyn/breaking_news.rss --output echo --args '
↪{item.title}'
```

Saving feed items to a Maildir folder:

```
feed2exec add "NASA breaking news" https://www.nasa.gov/rss/dyn/breaking_news.rss --
↪folder nasa
feed2exec fetch
```

This creates the equivalent of this configuration file in `~/.config/feed2exec.ini`:

```
[DEFAULT]
output = feed2exec.plugins.maildir
mailbox = '~/Maildir'

[NASA breaking news]
folder = nasa
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
```

Send new feed items to Transmission:

```
feed2exec add "Example torrent list" http://example.com/torrents/feed --output␣
↪transmission --folder /srv/incoming
```

Send new feed items to Mastodon, using the toot commandline client:

```
feed2exec add "My site" http://example.com/blog/feed --output exec --args 'toot post "
↪{item.title} {item.link}"'
```

Send new feed items to Twitter, using the tweet commandline client from python-twitter:

```
feed2exec add "My site on twitter" http://example.com/blog/feed --output exec --args
↪'tweet "{item.title:40s} {item.link:100s}"'
```

Show feed contents:

```
feed2exec add "NASA breaking news" https://www.nasa.gov/rss/dyn/breaking_news.rss --
↪output echo --args "{item.title} {item.link}"
feed2exec fetch
```

Multiple feeds can also be added with the OPML import command. See the *feed2exec manual page* document for more information including known issues and limitations.

CHAPTER 2

Installation

This can be installed using the normal Python procedures:

```
pip install feed2exec
```

It can also be installed from source, using:

```
pip install .
```

It can also be ran straight from the source, using:

```
python -m feed2exec
```

Important: feed2exec is explicitly written for Python 3. It may be possible to backport it to Python 2 if there is sufficient demand, but there are too many convenient Python3 constructs to make this useful. Furthermore, all dependencies are well-packaged for Py3 and the platform is widely available. Upgrade already.

The program may also be available as an official package from your Linux distribution.

Source, documentation and issues are available on GitLab.

Note: feed2exec relies on the feedparser module to parse feeds and as such has all the bugs and limitations of that modules. In particular, feeds with non-standard dates will break the parser, unless the dateparser module is installed.

CHAPTER 3

Why the name?

There are already feed2tweet and feed2imap out there so I figured I would just reuse the prefix and extend *both* programs at once.

Contents:

3.1 feed2exec manual page

3.1.1 Synopsis

feed2exec {add,ls,rm,fetch,import,export}

3.1.2 Description

This command will take a configured set of feeds and fire specific plugin for every new item found in the feed.

3.1.3 Options

--version	Show the version and exit.
--loglevel	choose specific log level [default: WARNING]
-v, --verbose	show what is happening (loglevel: VERBOSE)
-d, --debug	show debugging information (loglevel: DEBUG)
--syslog LEVEL	send LEVEL logs to syslog
--config TEXT	use a different configuration file
--database DB	use a different database
-h, --help	Show this message and exit.

3.1.4 Examples

Simple run with no side effects:

```
feed2exec parse https://www.nasa.gov/rss/dyn/breaking_news.rss --output echo --args '
↪{item.title}'
```

Saving feed items to a Maildir folder:

```
feed2exec add "NASA breaking news" https://www.nasa.gov/rss/dyn/breaking_news.rss --
↪folder nasa
feed2exec fetch
```

This creates the equivalent of this configuration file in `~/.config/feed2exec.ini`:

```
[DEFAULT]
output = feed2exec.plugins.maildir
mailbox = '~/Maildir'

[NASA breaking news]
folder = nasa
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
```

Send new feed items to Transmission:

```
feed2exec add "Example torrent list" http://example.com/torrents/feed --output␣
↪transmission --folder /srv/incoming
```

Send new feed items to Mastodon, using the toot commandline client:

```
feed2exec add "My site" http://example.com/blog/feed --output exec --args 'toot post "
↪{item.title} {item.link}"'
```

Send new feed items to Twitter, using the tweet commandline client from python-twitter:

```
feed2exec add "My site on twitter" http://example.com/blog/feed --output exec --args
↪'tweet "{item.title:40s} {item.link:100s}"'
```

Show feed contents:

```
feed2exec add "NASA breaking news" https://www.nasa.gov/rss/dyn/breaking_news.rss --
↪output echo --args "{item.title} {item.link}"
feed2exec fetch
```

3.1.5 Commands

- *parse*
- *fetch*
- *add*
- *ls*
- *rm*

- *import*
- *export*

parse

Usage:

```
parse URL
    [--output PLUGIN [--args ARG [ARG [...]]]
    [--filter PLUGIN] [--filter_args ARG [ARG [...]]]
    [--mailbox PATH] [--folder PATH]
```

The parse command loads and parses a single feed, without touching the database. This is similar to calling *add* then *fetch* on a single feed, but the feed is not kept in the configuration. This is designed to make quick tests with a new feed. The arguments are the same as the *add* command.

fetch

Usage:

```
fetch [--parallel | -p | --jobs N | -j N] [--force | -f] [--pattern pattern]
```

The fetch command iterates through all the configured feeds or those matching the `pattern` substring if provided.

Options:

--pattern TEXT	only fetch feeds matchin name or URL
--parallel	parse feeds in the background to improve performance
-j, --jobs N	start N jobs in parallel, implies `--parallel` which defaults to the number of CPUs detected on the machine
-f, --force	skip reading and writing the cache and will consider all entries as new
-n, --catchup	tell output plugins plugins to simulate their actions

add

Usage:

```
add NAME URL
    [--output PLUGIN [--args ARG [ARG [...]]]
    [--filter PLUGIN] [--filter_args ARG [ARG [...]]]
    [--mailbox PATH] [--folder PATH]
```

The add command adds the given feed `NAME` that will be fetched from the provided `URL`.

Options:

--output PLUGIN	use PLUGIN as an output module. defaults to `maildir` to store in a mailbox. use `null` or `echo` to just fetch the feed without doing anything. Modules are searched in the *feed2exec.plugins* package unless the name contains a dot in which case the whole Python search path is used.

--args ARGS	pass arguments ARGS to the output plugin. supports interpolation of feed parameters using, for example `{title}`
--filter PLUGIN	filter feed items through the PLUGIN filter plugin
--filter_args ARGS	arguments passed to the filter plugin
--mailbox PATH	folder to store email into, defaults to `~/Maildir`.
--folder PATH	subfolder to store the email into

Those parameters are documented more extensively in their equivalent settings in the configuration file, see below.

ls

The `ls` command lists all configured feeds as JSON packets.

rm

Usage:

```
rm NAME
```

Remove the feed named `NAME` from the configuration.

import

Usage:

```
import PATH
```

Import feeds from the file named PATH. The file is expected to have `outline` elements and only the `title` and `xmlUrl` elements are imported, as `NAME` and `URL` parameters, respectively.

export

Usage:

```
export PATH
```

Export feeds into the file named PATH. The file will use the feed NAME elements as `title` and the URL as `xmlUrl`.

3.1.6 Files

Configuration file

The configuration file is loaded from (and written to, by `add`) `~/.config/feed2exec.ini` or `$XDG_CONFIG_HOME/feed2exec.ini`. It can also be specified with the `--config` commandline parameter. This is an example configuration snippet:

```
[NASA breaking news]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
output = feed2exec.plugins.echo
args = {title} {link}
```

Naturally, those settings can be changed directly in the config file. Note that there is a `[DEFAULT]` section that can be used to apply settings to all feeds. For example, this will make all feeds store new items in a maildir subfolder:

```
[DEFAULT]
output = feed2exec.plugins.maildir
folder = feeds
```

This way individual feeds do not need to be individually configured.

Note: feed2exec does not take care of adding the folder to "subscriptions" in the mailbox. it is assumed that folders are auto-susbcribed or the user ignores subscription. if that is a problem, you should subscribe to the folder by hand in your email client when you add a new config. you can also subscribe to a folder (say `feeds` above) directly using the `doveadm mailbox subscribe feeds` command in Dovecot, for example.

The following configuration parameters are supported:

name Human readable name for the feed. Equivalent to the `NAME` argument in the `add` command.

url Address to fetch the feed from. Can be HTTP or HTTPS, but also `file://` resources for test purposes.

output Output plugin to use. Equivalent to the `--output` option in the `add` command.

args Arguments to pass to the output plugin. Equivalent to the `--args` option in the `add` command.

filter Filter plugin to use. Equivalent to the `--filter` option in the `add` command.

mailbox Store emails in that mailbox prefix. Defaults to `~/Maildir`.

folder Subfolder to use when writing to a mailbox. By default, a *slugified* version of the feed name (where spaces and special character are replaced by -) is used. For example, the feed named "NASA breaking news" would be stored in `~/Maildir/nasa-breaking-news/`.

catchup Skip to the latest feed items. The feed is still read and parsed, and new feed items are added to the database, but output plugins are never called.

pause Completely skip feed during fetch or parse. Similar to catchup, but doesn't fetch the feed at all and doesn't touch the cache.

Here is a more complete example configuration with all the settings used:

```
# this section will apply to all feeds
[DEFAULT]
# special folder location for maildir. I use this when I have multiple
# accounts synchronized with Offlineimap
mailbox = ~/Maildir/Remote/

# a feed to store NASA breaking news entry in a "nasa" subfolder
# this also demonstrates the droptitle filter
[NASA breaking news]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
folder = nasa
filter = feed2exec.plugins.droptitle
filter_args = trump

# some maildir storage require dots to get subfolders. for example,
# this will store messages in INBOX/feeds/images/ on Dovecot
[NASA image of the day]
url = https://www.nasa.gov/rss/dyn/lg_image_of_the_day.rss
```

(continues on next page)

(continued from previous page)

```
folder = .feeds.images

# same feed, but save to wayback machine
[NASA IOTD wayback]
url = https://www.nasa.gov/rss/dyn/lg_image_of_the_day.rss
output = feed2exec.plugins.wayback

# this demonstrates the emptysummary filter, which fixes GitHub feeds
# that lack a proper summary
[restic]
url = https://github.com/restic/restic/tags.atom
filter = feed2exec.plugins.emptysummary

# saving to a mbox folder, one file per feed instead of one file per item
[International Space Station Reports]
url = http://blogs.nasa.gov/stationreport/feed/
mailbox = ~/Mail/
folder = stationreport.mbx

# simple generic exec call example: check for broken links using linkchecker
[NASA linkchecker]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
output = feed2exec.plugins.exec
args = linkchecker --check-extern --no-robots --recursion-level 1 --quiet '{item.link}
↪'

# same, but with a Ikiwiki RSS feed, which needs fixing
[Ikiwiki linkchecker]
url = http://ikiwiki.info/recentchanges/index.rss
output = feed2exec.plugins.exec
filter = feed2exec.plugins.ikiwiki_recentchanges
args = linkchecker --check-extern --no-robots --recursion-level 1 --quiet '{item.link}
↪'

# retweet hurricane news
[NASA Hurricane breaking news]
url = https://www.nasa.gov/rss/dyn/hurricaneupdate.rss
output = feed2exec.plugins.exec
args = tweet "{item.title:.40s} {item.link:.100s}"

# same, but on the mastodon network
#
# we can have multiple entries with the same URL without problems, as
# long as the feed name is different. it does mean that the feed will
# be fetched and parsed multiple times, unfortunately.
#
# this could be improved to include the '{item.summary}' and extra markup,
# for example.
[NASA Hurricane breaking news - Mastodon]
url = https://www.nasa.gov/rss/dyn/hurricaneupdate.rss
output = feed2exec.plugins.exec
# unfortunately, this will noisily report the URL of the posted link,
# which you may not want. to avoid that, encourage upstream to do the
# right thing: https://github.com/ihabunek/toot/issues/46 ... or use
# another tool listed here:
# https://github.com/tootsuite/documentation/blob/master/Using-Mastodon/Apps.md
args = toot post "{item.title} {item.link}"
```

(continues on next page)

(continued from previous page)

```
# output is disabled here. feed will be fetched and parsed, but no
# toot will be sent
catchup = True

# same, but on the Pump.io network
[NASA Hurricane breaking news - Pump]
url = https://www.nasa.gov/rss/dyn/hurricaneupdate.rss
output = feed2exec.plugins.exec
args = p post note "{item.title} {item.link}"

# crude podcast client
[NASA Whats up?]
url = https://www.nasa.gov/rss/dyn/whats_up.rss
output = feed2exec.plugins.exec
# XXX: this doesn't handle errors properly: if there is a feed without
# enclosures, the whole thing will crash.
args = wget -P /srv/podcasts/nasa/ "{item.enclosures[0].href}"
# feed is paused here. feed will not be fetched and parsed at all and
# no post will be sent.
pause = True

# download torrents linked from a RSS feed
[torrents]
url = http://example.com/torrents.rss
output = feed2exec.plugins.exec
args = transmission-remote -a '{item.link}' -w '/srv/incoming'

# same thing with an actual plugin
[torrents]
url = http://example.com/torrents.rss
output = feed2exec.plugins.transmission
args = seedbox.example.com
folder = /srv/incoming
```

Cache database

The feeds cache is stored in a `feed2exec.db` file. It is a SQLite database and can be inspected using standard sqlite tools. It is used to keep track of which feed items have been processed. To clear the cache, you can simply remove the file, which will make the program process all feeds items from scratch again. In this case, you should use the `--catchup` argument to avoid duplicate processing. You can also use the `null` output plugin to the same effect.

3.1.7 Limitations

Feed support is only as good as `feedparser` library which isn't as solid as I expected. In particular, I had issues with feeds without dates and without guid.

Unit test coverage is incomplete, but still pretty decent, above 90%.

The `exec` plugin itself is not well tested and may have serious security issues.

API, commandline interface, configuration file syntax and database format can be changed until the 1.0 release is published, at which point normal Semantic Versioning semantics apply.

The program is written mainly targeting Python 3.5 and should work in 3.6. Python 2.7 is not supported anymore.

The SQL storage layer is badly written and is known to trigger locking issues with SQLite when doing multiprocessing. The global LOCK object could be used to work around this issue but that could mean pretty bad coupling. A good inspiration may be the beets story about this problem. And of course, another alternative would be to considering something like SQLalchemy instead of rolling our own ORM.

Older feed items are not purged from the database when they disappear from the feed, which may lead to database bloat in the long term. Similarly, there is no way for plugins to remove old entry that expire from the feed.

3.1.8 See also

feed2exec-plugins(1), *feed2imap(1)*, *rss2email(1)*

3.2 Design

This is a quick prototype that turned out to be quite usable. The design is minimal: some home-made ORM for the feed storage, crude parallelism with the `multiprocessing` module and a simple plugin API using `importlib`.

More information about known issues and limitations in the *feed2exec manual page* document.

3.2.1 Quick tour

The most common workflow is through the *fetch* subcommand and goes something like this:

1. `__main__.py` is the main entrypoint, managed through the `click` module, which normally calls function defined in `feeds.py`. `feed2exec.feeds.main()` creates a *feed2exec.feeds.FeedManager* object which gets passed to subcommands. In our case, it passes the control to the `fetch` subcommand.
2. The fetch command calls the *feed2exec.feeds.FeedManager.fetch()* function which creates a *feed2exec.feeds.Feed* object that is then used to parse the feed and return it as an opaque *data* object as returned by `feedparser`.
3. `fetch` then calls the *feed2exec.feeds.FeedManager.dispatch()* function that calls the various filter and output plugins, passing in the feed configuration and one item at a time. The filters can modify the feed items while the output plugins are responsible for writing them somewhere. That distinction is mostly arbitrary, but the return values of the output plugins matter, while filters do not.

The feed cache is stored in a minimal `sqlite3` database. That database could be extended to keep the body of the feed and other data to enable more powerful features, but I haven't had the need for this yet.

Configuration is stored in a `.ini` file or whatever `configparser` supports. It was originally stored in the database as well, but it was found inconvenient to modify by hand and a configuration file was used instead. The `.ini` file format was chosen because it is well supported by Python and allows for default settings.

There is the possibility for this project to cover more than RSS/Atom feeds. In theory, the `parse` function could also be pluggable and support *reading* from other data sources like Twitter or Facebook, which would bring us closer to the IFTTT concept.

3.2.2 Plugin system

Plugins are documented in the *Plugins* section. You can also refer to the *Writing new plugins* section if you wish to write a new plugin or extend an existing one.

The plugin system uses a simple `importlib` based architecture where plugin are simple Python modules loaded at runtime based on a module path provided by the user. This pattern was inspired by a StackOverflow discussion.

The following options were also considered:

- pluggy: used by py.test, tox and devpi
- yapsy
- PluginBase
- plugnplay
- click-plugins: relevant only to add new commands
- PyPA plugin discovery

Those options were ultimately not used because they add an aditionnal dependency and are more complicated than a simple `import`. We also did not need plugin listing or discovery, which greatly simplifies our design.

There is some code duplication between different parts (e.g. the *`feed2exec.plugins.output()`* and *`feed2exec.plugins.filter()`* plugin interfaces, the `maildir` and `mbox` plugins, etc), but never more than twice.

3.2.3 Concurrent processing

The threading design may be a little clunky and is certainly less tested, which is why it is disabled by default (use `--parallel` to use it). There are known deadlocks issues with high concurrency scenarios (e.g. with `catchup` enabled).

I had multiple design in minds: the current one (`multiprocessing.Pool` and `pool.apply_async`) vs `aiohttp` (on the `asyncio` branch) vs `pool.map` (on the `threadpoolmap` branch). The `aiohttp` design was very hard to diagnose and debug, which made me abandon the whole thing. After reading up on Curio and Trio, I'm tempted to give async/await a try again, but that would mean completely dropping 2.7 compatibility. The `pool.map` design is just badly adapted, as it would load all the feed's datastructure in memory before processing them.

3.2.4 Test suite

The test suite is in `feed2exec/tests` but also as doctest comments in some functions imported from the ecdysis project. You can run all the tests with pytest, using, for example:

```
pytest-3
```

This is also hooked into the `setup.py` command, so this also works:

```
python3 setup.py test
```

Note: This alias is called by `tox` in the `tox.ini` file through the `pytest-runner` plugin. It is defined in the `setup.cfg` file. `tox`, in turn, gets called by the Continuous Integration (CI) system through the `.gitlab-ci.yml` file.

Enabling the catchlog plugin will also enable logging in the test suite which will help diagnostics.

Note that some tests will fail in Python 2, as the code is written and tested in Python3. Furthermore, the feed output is taken from an up to date (5.2.1) feedparser version, so the tests are marked as expected to fail for lower versions. You should, naturally, run and write tests before submitting patches. See the *Writing tests* section for more information about how to write tests.

The test suite also uses the betamax module to cache HTTP requests locally so the test suite can run offline. If a new test requires networking, you can simply add a new test doing requests with the right fixture (`feed2exec.tests.fixtures.betamax()`), and a new recording will be added to the source tree. Note that you can also use the normal `betamax_session()` fixture provided upstream if you are going to do standalone HTTP request (not going through the feed2exec libraries). If a new test is added in an *existing* test, you may need to configure recording (in `feed2exec/tests/conftest.py`) to `new_episodes`:

```
config.default_cassette_options['record_mode'] = 'none'
```

We commit the recordings in git so the test suite actually runs offline, so be careful about the content added there. Ideally, the license of that content should be documented in `debian/copyright`.

vcrpy was first used for tests since it was simpler and didn't require using a global `requests.session.Session` object. But in the end betamax seems better maintained and more flexible: it supports pytest fixtures, for example, and multiple cassette storage (including vcr backwards compatibility). Configuration is also easier, done in `feed2exec/tests/conftest.py`. Using a session also allows us to use a custom user agent.

3.2.5 Comparison

`feed2exec` is a fairly new and minimal program, so features you may expect from another feed reader may not be present. I chose to write a new program because, when I started, both existing alternatives were in a questionable state: feed2imap was mostly abandoned and rss2email's maintainer was also unresponsive. Both were missing the features I was looking for, which was to unify my feed parsers in a single program: i needed something that could deliver mail, run commands and send tweets. The latter isn't done yet, but I am hoping to complete this eventually.

The program may not be for everyone, however, so I made those comparison tables to clarify what feed2exec does compared to the alternatives.

General information:

Program	Version	Date	SLOC	Language
feed2exec	0.10	2017	989	Python
feed2imap	1.2.5	2015	3238	Ruby
rss2email	3.9	2014	1754	Python

- version: the version analysed
- date: the date of that release
- SLOC: Source Lines of Codes as counted by sloccount, only counting dominant language (e.g. excluding XML from test feeds) and excluding tests
- Language: primary programming language

Delivery options:

Program	Maildir	Mbox	IMAP	SMTP	sendmail	exec
feed2exec	✓	✓				✓
feed2imap	✓		✓			
rss2email			✓	✓	✓	

- maildir: writing to Maildir folders. r2e has a pull request to implement maildir support, but it's not merged at the time of writing
- IMAP: sending emails to IMAP servers
- SMTP: delivering emails over the SMTP protocol, with authentication

- sendmail: delivering local using the local MTA
- exec: run arbitrary comands to run on new entries. feed2imap has a `execurl` parameter to execute commands, but it receives an unparsed dump of the feed instead of individual entries. rss2email has a postprocess filter that is a Python plugin that can act on indiviual (or digest) messages which could possibly be extended to support arbitrary commands, but that is rather difficult to implement for normal users.

Features:

Program	Pause	OPML	Retry	Images	Filter	Reply	Digest
feed2exec	✓	✓			✓	✓	
feed2imap		✓	✓	✓	✓		
rss2email	✓	✓	✓		✓	✓	✓

- pause: feed reading can be disabled temporarily by user. in feed2exec, this is implemented with the `pause` configuration setting. the `catchup` option can also be used to catchup with feed entries.
- retry: tolerate temporary errors. For example, `feed2imap` will report errors only after 10 failures.
- images: download images found in feed. `feed2imap` can download images and attach them to the email.
- filter: if we can apply arbitrary filters to the feed output. feed2imap can apply filters to the unparsed dump of the feed.
- reply: if the generated email 'from' header is usable to make a reply. `rss2email` has a `use-publisher-email` setting (off by default) for this, for example. feed2exec does this by default.
- digest: possibility of sending a single email per run instead of one per entry

Note: `feed2imap` supports only importing OPML feeds, exporting is supported by a third-party plugin.

3.3 API documentation

This is the API documentation of the program. It should explain how to create new plugins and navigate the code.

3.3.1 Feeds module

This is the core modules that processes all feeds and takes care of the storage. It's where most of the logic lies.

fast feed parser that offloads tasks to plugins and commands

class `feed2exec.feeds.`**FeedManager**(*conf_path*, *db_path*, *pattern=None*)

a feed manager fetches and stores feeds.

this is a "controller" in a "model-view-controller" pattern. it derives the "model" (*feed2exec.feeds.FeedConfStorage*) for simplicity's sake, and there is no real "view" (except maybe *__main__*).

fetch(*parallel=False*, *force=False*, *catchup=False*)

main entry point for the feed fetch routines.

this iterates through all feeds configured in the parent *feed2exec.feeds.FeedConfStorage* that match the given `pattern`, fetches the feeds and dispatches the parsing, which in turn dispatches the plugins.

Parameters

- **pattern** (*str*) – restrict operations to feeds named `pattern`. passed to parent *feed2exec.feeds.FeedConfStorage* as is
- **parallel** (*bool*) – parse feeds in parallel, using `multiprocessing`
- **force** (*bool*) – force plugin execution even if entry was already seen. passed to `feed2exec.feeds.parse` as is
- **catchup** (*bool*) – set the *catchup* flag on the feed, so that output plugins can avoid doing any permanent changes.

dispatch(*feed*, *data*, *lock=None*, *force=False*)
process parsed entries and execute plugins

This handles locking, caching, and filter and output plugins.

opml_import(*opmlfile*)
import a file stream as an OPML feed in the feed storage

class `feed2exec.feeds.`**Feed**(*name*, **args*, ***kwargs*)
basic data structure representing a RSS or Atom feed.

it derives from the base `feedparser.FeedParserDict` but forces the element to have a `name`, which is the unique name for that feed in the *feed2exec.feeds.FeedManager*. We also add convenience functions to parse (in parallel) and normalize feed items.

on intialization, a new `requests.Session` object is created to be used across all requests. therefore, as long as a first FeedManager() object was created, FeedManager._session can be used by plugins.

For all intents and purposes, this can be considered like a dict() unless otherwise noted.

session
the session property

static sessionConfig(*session*)
our custom session configuration

we change the user agent and set the file:// hanlder. extra configuration may be performed in the future and will override your changes.

this can be used to configure sessions used externally, for example by plugins.

normalize(*item=None*)
normalize feeds a little more than what feedparser provides.

we do the following operation:

1. add more defaults to item dates (issue #113)
2. missing GUID in some feeds (issue #112)
3. link normalization fails on some feeds, particilarly GitHub, where feeds are /foo instead of https://github.com/foo. unreported for now.

parse(*body*)
parse the body of the feed

this parses the given body using `feedparser` and calls the plugins configured in the `feed` (using *feed2exec.plugins.output()* and *feed2exec.plugins.filter()*). updates the cache with the found items if the `output` plugin succeeds (returns True) and if the `filter` plugin doesn't set the `skip` element in the feed item.

Todo this could be moved to a plugin, but then we'd need to take out the cache checking logic, which would remove most of the code here...

Parameters

- **body** (*bytes*) – the body of the feed, as returned by :func:fetch
- **self** (*dict*) – a feed object used to pass to plugins and debugging
- **lock** (*object*) – a `multiprocessing.Lock` object previously initialized. if None, the global *LOCK* variable will be used: this is used in the test suite to avoid having to pass locks all the way through the API. this lock is in turn passed to plugin calls.
- **force** (*bool*) – force plugin execution even if entry was already seen. passed to `feed2exec.feeds.parse` as is

Return dict the parsed data

fetch()
fetch the feed content and return the body, in binary

This will call `logging.warning()` for exceptions `requests.exceptions.Timeout` and `requests.exceptions.ConnectionError` as they are transient errors and the user may want to ignore those.

Other exceptions raised from `requests.exceptions` (like TooManyRedirects or HTTPError but basically any other exception) may be a configuration error or a more permanent failure so will be signaled with `logging.error()`.

this will return the body on success or None on failure

class feed2exec.feeds.**FeedConfStorage**(*path*, *pattern=None*)
Feed configuration stored in a config file.

This derives from `configparser.RawConfigParser` and uses the `.ini` file set in the `path` member to read and write settings.

Changes are committed immediately, and no locking is performed so loading here should be safe but not editing.

The particular thing about this configuration is that there is an iterator that will yield entries matching the `pattern` substring provided in the constructor.

add(*name*, *url*, *output=None*, *args=None*, *filter=None*, *filter_args=None*, *folder=None*, *mailbox=None*)
add the designated feed to the configuration

this is not thread-safe.

set(*section*, *option*, *value=None*)
override parent to make sure we immediately write changes

not thread-safe

remove_option(*section*, *option*)
override parent to make sure we immediately write changes

not thread-safe

remove(*name*)
convenient alias for `configparser.RawConfigParser.remove_section()`

not thread-safe

commit()
write the feed configuration

see `configparser.RawConfigParser.write()`

3.3.2 Main entry point

The main entry point of the program is in the *feed2exec.__main__* module. This is to make it possible to call the program directly from the source code through the Python interpreter with:

```
python -m feed2exec
```

All this code is here rather than in `__init__.py` to avoid requiring too many dependencies in the base module, which contains useful metadata for `setup.py`.

This uses the `click` module to define the base command and options.

fast feed parser that offloads tasks to plugins and commands

3.3.3 Plugins

Plugin interface

In this context, a "plugin" is simply a Python module with a defined interface.

`feed2exec.plugins.`**`output`**(*feed*, *item*, *lock=None*)

load and run the given plugin with the given arguments

an "output plugin" is a simple Python module with an `output` callable defined which will process arguments and should output them somewhere, for example by email or through another command. the plugin is called (from `feed2exec.feeds.parse()`) when a new item is found, unless cache is flushed or ignored.

The "callable" can be a class, in which case only the constructor is called or a function. The `*args` and `**kwargs` parameter SHOULD be used in the function definition for forward-compatibility (ie. to make sure new parameters added do not cause a regression).

Plugins should also expect to be called in parallel and should use the provided `lock` (a multiprocessor.Lock object) to acquire and release locks around contentious resources.

The following keywords are usually replaced in the arguments:

- {item.link}
- {item.title}
- {item.description}
- {item.published}
- {item.updated}
- {item.guid}

The full list of such parameters is determined by the :module:feedparser module.

Similarly, feed parameters from the configuration file are accessible.

> **Caution:** None of those parameters are sanitized in any way other than what feedparser does, so plugins writing files, executing code or talking to the network should be careful to sanitize the input appropriately.

The feed and items are also passed to the plugin as keyword arguments. Plugins should especially respect the `catchup` argument that, when set, forbids plugins to do any permanent activity. For example, plugins MUST NOT run commands, write files, or make network requests. In general, "catchup mode" should be *fast*: it allows users to quickly catchup with new feeds without firing plugins, but it should *also* allow users to test

configurations so plugins SHOULD give information to the user about what would have been done by the plugin without `catchup`.

Parameters

- **feed** (*dict*) – the feed metadata
- **item** (*dict*) – the updated item

Return object the loaded plugin

Note: more information about plugin design is in the *Writing new plugins* document.

`feed2exec.plugins.`**`filter`**(*feed*, *item*, *lock=None*)
call filter plugins.

very similar to the output plugin, but just calls the `filter` module member instead of `output`

Todo: common code with output() should be factored out, but output() takes arguments...

`feed2exec.plugins.`**`resolve`**(*plugin*)
resolve a short plugin name to a loadable plugin path

Some parts of feed2exec allow shorter plugin names. For example, on the commandline, users can pass *maildir* instead of *feed2exec.plugins.maildir*.

Plugin resolution works like this:

1. search for the module in the *feed2exec.plugins* namespace
2. if that fails, consider the module to be an absolute path

Note: actual plugins are documented in the *Plugins* document.

3.3.4 Utilities

Those are various utilities reused in multiple modules that did not fit anywhere else.

various reusable utilities

`feed2exec.utils.`**`slug`**(*text*)
Make a URL-safe, human-readable version of the given text

This will do the following:

1. decode unicode characters into ASCII
2. shift everything to lowercase
3. strip whitespace
4. replace other non-word characters with dashes
5. strip extra dashes

This somewhat duplicates the `Google.slugify()` function but slugify is not as generic as this one, which can be reused elsewhere.

```
>>> slug('test')
'test'
>>> slug('Mørdag')
'mordag'
>>> slug("l'été c'est fait pour jouer")
'l-ete-c-est-fait-pour-jouer'
>>> slug(u"çafe au lait (boisson)")
'cafe-au-lait-boisson'
>>> slug(u"Multiple  spaces -- and symbols! -- merged")
'multiple-spaces-and-symbols-merged'
```

This is a simpler, one-liner version of the slugify module.

taken from ecdysis

feed2exec.utils.**make_dirs_helper**(*path*)

Create the directory if it does not exist

Return True if the directory was created, false if it was already present, throw an OSError exception if it cannot be created

```
>>> import tempfile
>>> import os
>>> import os.path as p
>>> d = tempfile.mkdtemp()
>>> make_dirs_helper(p.join(d, 'foo'))
True
>>> make_dirs_helper(p.join(d, 'foo'))
False
>>> make_dirs_helper('')
False
>>> make_dirs_helper(p.join('/dev/null', 'foo')) # doctest: +ELLIPSIS
Traceback (most recent call last):
    ...
NotADirectoryError: [Errno 20] Not a directory: ...
>>> os.rmdir(p.join(d, 'foo'))
>>> os.rmdir(d)
>>>
```

feed2exec.utils.**find_test_file**(*name='.'*)

need to be updated from ecdysis

feed2exec.utils.**find_parent_module**()

find the name of a the first module calling this module

if we cannot find it, we return the current module's name (__name__) instead.

taken from ecdysis

3.4 Plugins

This is a quick overview of the available plugins.

3.4.1 Output plugins

Archive

feed2exec.plugins.archive.**DEFAULT_ARCHIVE_DIR = '/run/user/1000/'**
default archive directory

feed2exec.plugins.archive.**output**(**args*, *feed=None*, *item=None*, ***kwargs*)
The archive plugin saves the feed's item.link URLs into a directory, specified by DEFAULT_ARCHIVE_DIR or through the output *args* value.

Example:

```
[NASA breaking news]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
output = archive
args = /srv/archive/nasa/
```

The above will save the "NASA breaking news" into the `/srv/archive/nasa` directory. Do *not* use interpolation here as the feed's variable could be used to mount a directory transversal attack.

Echo

class feed2exec.plugins.echo.**output**(**args*, *feed=None*, ***kwargs*)
This plugin outputs, to standard output, the arguments it receives. It can be useful to test your configuration. It also creates a side effect for the test suite to determine if the plugin was called.

This plugin does a similar thing when acting as a filter.

feed2exec.plugins.echo.**filter**
This filter just keeps the feed unmodified. It is just there for testing purposes.

alias of *feed2exec.plugins.echo.output*

Error

feed2exec.plugins.error.**output**(**args*, ***kwargs*)
The error plugin is a simple plugin which raises an exception when called. It is designed for use in the test suite and should generally not be used elsewhere.

Exec

feed2exec.plugins.exec.**output**(*command*, **args*, *feed=None*, ***kwargs*)
The exec plugin is the ultimate security disaster. It simply executes whatever you feed it without any sort of sanitization. It does avoid to call to the shell and executes the command directly, however. Feed contents are also somewhat sanitized by the feedparser module, see the Sanitization documentation for more information in that regard. That is limited to stripping out hostile HTML tags, however.

You should be careful when sending arbitrary parameters to other programs. Even if we do not use the shell to execute the program, an hostile feed could still inject commandline flags to change the program behavior without injecting shell commands themselves.

For example, if a program can write files with the `-o` option, a feed could set their title to `-oevil` to overwrite the `evil` file. The only way to workaround that issue is to carefully craft the commandline so that this cannot happen.

Alternatively, writing a Python plugin is much safer as you can sanitize the arguments yourself.

Example:

```
[NASA Whats up?]
url = https://www.nasa.gov/rss/dyn/whats_up.rss
output = feed2exec.plugins.exec
args = wget -P /srv/archives/nasa/ {item.link}
```

The above is the equivalent of the archive plugin: it will save feed item links to the given directory.

Maildir

class feed2exec.plugins.maildir.**output**(*to_addr=None*, *feed=None*, *item=None*, *lock=None*, **args*, ***kwargs*)

The maildir plugin will save a feed item into a Maildir folder.

The configuration is a little clunky, but it should be safe against hostile feeds.

Parameters **to_addr** (*str*) – the email to use as "to" (defaults to USER@localdomain)

Example:

```
[NASA breaking news]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
mailbox = ~/Maildir/
folder = nasa
args = me@example.com
```

The above will save new feed items from the NASA feed into the ~/Maildir/nasa/ maildir folder, and will set the *To* field of the email to *me@example.com*.

Mbox

class feed2exec.plugins.mbox.**output**(*to_addr=None*, *feed=None*, *item=None*, *lock=None*, **args*, ***kwargs*)

The mbox plugin will save a feed item in a Mbox mailbox.

This is mostly for testing purposes, but can of course be used in the unlikely event where you prefer mbox folders over the *feed2exec.plugins.maildir* plugin.

Parameters **to_addr** (*str*) – the email to use as "to" (defaults to USER@localdomain)

Todo There is some overlap between the code here and the maildir implementation. Refactoring may be in order, particularly if we add another mailbox format, though that is unlikely.

Null

feed2exec.plugins.null.**output**(**args*, ***kwargs*)

This plugin does nothing. It can be useful in cases where you want to catchup with imported feeds.

feed2exec.plugins.null.**filter**(*item=None*, **args*, ***kwargs*)

The null filter removes all elements from a feed item

Transmission

feed2exec.plugins.transmission.**sanitize**(*text*, *repl='-'*)

like utils.slug, but without lowercase and allow custom replacement

```
>>> sanitize('test')
'test'
>>> sanitize('../../../etc/password')
'etc-password'
>>> sanitize('Foo./.bar', repl='.')
'Foo.bar'
```

feed2exec.plugins.transmission.**output**(*hostname='localhost'*, **args*, *feed=None*, *item=None*, ***kwargs*)

the transmission plugin will send feed items to a transmission instance

it assumes the `transmission-remote` command is already installed and configured to talk with transmission.

the hostname is passed in the `args` configuration and defaults to localhost. the `folder` parameter is also used to determine where to save the actual torrents files.

note that this will also append a sanitized version of the item title, if a folder is provided. this is to allow saving series in the same folder.

if the title is unique for each torrent, you may use a filter to set the title to the right location.

Wayback

feed2exec.plugins.wayback.**output**(**args*, *feed=None*, *item=None*, ***kwargs*)

This plugin saves the feed items *link* element to the wayback machine. It will retry URLs that fail, so it may be necessary to manually catchup feeds if they have broken *link* fields.

Example:

```
[NASA IOTD wayback]
url = https://www.nasa.gov/rss/dyn/lg_image_of_the_day.rss
output = feed2exec.plugins.wayback
```

The above will save the Image of the day updates to the wayback machine.

3.4.2 Filter plugins

Droptitle

feed2exec.plugins.droptitle.**filter**(**args*, *feed=None*, *item=None*, ***kwargs*)

the droptitle filter will drop any feed item with a title matching the given args.

Example:

```
[NASA breaking news]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
filter = feed2exec.plugins.droptitle
filter_args = Trump
```

The above will process the feed items according to the global configuration, but will skip any item that has the word "Trump" anywhere in the title field.

Emptysummary

feed2exec.plugins.emptysummary.**filter**(**args*, *feed=None*, *item=None*, ***kwargs*)
example of fixes for a broken feed, in this case, the GitHub release feed which (sometimes) sends empty contents, in which case the item link field is used as a summary instead.

Html2text

class feed2exec.plugins.html2text.**filter**(**args*, *feed=None*, *item=None*, ***kwargs*)
This filter plugin takes a given feed item and adds a `content_plain` field with the HTML parsed as text.

Important: the html2text plugin is called automatically from the email output plugins and should normally not be called directly.

---

static parse(*html=None*)
parse html to text according to our preferences. this is where subclasses can override the HTML2Text settings or use a completely different parser

Ikiwiki Recentchanges

feed2exec.plugins.ikiwiki_recentchanges.**filter**(**args*, *item=None*, ***kwargs*)
the ikiwiki_recentchanges plugin fixes links in ikiwiki feeds

Ikiwiki recent changes show all the recent edits to pages, but the `<link>` element doesn't point to the edit page: it points to the recent changes page itself, which make them useless for link checking or archival purposes.

This parses the recent changes entries and extracts the relevant links from it.

An alternative to this is to use the following entry to generate a special feed in Ikiwiki:

```
[[!inline pages="*" feeds=yes feedonly=yes feedfile=archive show=10]]
```

This generates a feed with proper `<link>` elements but requires write access to the wiki.

This will also add the date to the URL GUID so that we refresh when a page is updated. Otherwise feed2exec would think the entry has already been passed.

3.4.3 Writing new plugins

Most of the actual work in the program is performed by plugins. A plugin is a simple Python module that has a `output` or `filter` "callable" (function or class) with a predefined interface.

Basic plugin principles

To write a new plugin, you should start by creating a simple Python module, in your PYTHONPATH. You can find which directories are in the path by calling:

```
$ python3 -c "import sys; print(sys.path)"
['', '/usr/lib/python35.zip', '/usr/lib/python3.5', '/usr/lib/python3.5/plat-x86_64-
↪linux-gnu', '/usr/lib/python3.5/lib-dynload', '/usr/local/lib/python3.5/dist-
↪packages', '/usr/lib/python3/dist-packages']
```

In the above example, a good location would be `/usr/local/lib/python3.5/dist-packages`. The naming convention is loose: as long as the plugin matches the expected API, it should just work. For the purpose of this demonstration, we'll call our plugin trumpery, so we will create the plugin code like this:

```
touch /usr/local/lib/python3.5/dist-packages/trumpery.py
```

Naturally, if you are going to write multiple plugins, you may want to regroup your multiple plugins in a package, see the module documentation for more information about this concept in Python.

Note: There is a rudimentary plugin resolution process that looks for plugins first in the *feed2exec.plugins* namespace but then globally. This is done in *feed2exec.plugins.resolve()*, called from the `add` and `parse` commands. This means that the absolute path is expected to be used in the configuration file and internally.

You are welcome to distribute plugins separately or send them as merge requests, see *Contribution guide* for more information on how to participate in this project. We of course welcome contributions to this documentation as well!

Filters

Now, you need your plugin to do something. In our case, let's say we'd like to skip any feed entry that has the word Trump in it. For that purpose, we'll create a plugin similar to the already existing *feed2exec.plugins.droptitle* plugin, but that operates on the *body* of the feed, but that also hardcodes the word, because this is just a demonstration and we want to keep it simple. Let's look at the title plugin to see how it works:

```
def filter(*args, feed=None, item=None, **kwargs):
    '''the droptitle filter will drop any feed item with a title matching
    the given args.

    Example::

      [NASA breaking news]
      url = https://www.nasa.gov/rss/dyn/breaking_news.rss
      filter = feed2exec.plugins.droptitle
      filter_args = Trump

    The above will process the feed items according to the global
    configuration, but will skip any item that has the word "Trump"
    anywhere in the title field.
    '''
    item['skip'] = ' '.join(args) in item.get('title', '')
```

That may look like complete gibberish to you if you are not familiar with programming or with Python programming in particular. But let's take this from the top and copy that in our own plugin. The first line declares a function that takes at least a `feed` and a `item` argument, but can also accept any other arbitrary argument. This is important because we want to have the plugin keep on working if the plugin API changes in the future. This is called "forward-compatibility". So let's copy that in our plugin and add a `pass` statement to make sure the plugin works (even if it does nothing for now):

```
def filter(*args, feed=None, item=None, **kwargs):
    pass
```

We can already test our plugin by adding it to our configuration, in `~/.config/feed2exec.ini`:

```
[NASA]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
```

(continues on next page)

(continued from previous page)

```
output = feed2exec.plugins.echo
args = {item.title}
filter = trumpery
```

Notice how we use the `output` plugin to show the title of feed items selected, as a debugging tool. Let's fetch this feed in debugging mode to see what happens:

```
$ python3 -m feed2exec --verbose fetch --force
opening local file /home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.xml
parsing feed file:///home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.
↪xml (10355 bytes)
connecting to database at ./doc/feed2exec.db
arguments received: ('President Trump Welcomes Home Record-breaking NASA Astronaut␣
↪Peggy Whitson',)
arguments received: ('Three International Space Station Crewmates Safely Return to␣
↪Earth',)
arguments received: ('NASA Statement on Nomination for Agency Administrator',)
arguments received: ('NASA Television to Air Return of Three International Space␣
↪Station Crew Members',)
arguments received: ('NASA and Iconic Museum Honor Voyager Spacecraft 40th Anniversary
↪',)
arguments received: ('NASA's Johnson Space Center Closes Through Labor Day for␣
↪Tropical Storm Harvey',)
arguments received: ('NASA Cancels Planned Media Availabilities with Astronauts',)
arguments received: ('NASA Awards $400,000 to Top Teams at Second Phase of 3D-
↪Printing Competition',)
arguments received: ('NASA Awards Contract for Center Protective Services for Glenn␣
↪Research Center',)
arguments received: ('NASA Announces Cassini End-of-Mission Media Activities',)
1 feeds processed
```

Good! The feed is fetched and items are displayed. It means our filter didn't interfere, but now it's time to make it *do* something. To skip items, we need to set the `skip` attribute for the feed item to *True* if we want to skip it and *False* otherwise. So we'll use a simple recipe, a bit like *droptitle* does, but simpler, to look at the feed content to look for our evil word. The `feedparser` documentation tells us feed items have a summary field which we can inspect. There's also a content list, but that's a little more complicated so we'll skip that for now. So, let's set the `skip` parameter to match if there is the evil word in our feed item, like this:

```
def filter(*args, feed=None, item=None, **kwargs):
    item['skip'] = 'Trump' in item.get('summary', '')
```

And let's see the result (note that we use the `--force` argument here otherwise we would just skip all items because of the cache):

```
$ python3 -m feed2exec --verbose fetch --force
opening local file /home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.xml
parsing feed file:///home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.
↪xml (10355 bytes)
connecting to database at ./doc/feed2exec.db
item President Trump Welcomes Home Record-breaking NASA Astronaut Peggy Whitson of␣
↪feed NASA filtered out
arguments received: ('Three International Space Station Crewmates Safely Return to␣
↪Earth',)
item NASA Statement on Nomination for Agency Administrator of feed NASA filtered out
arguments received: ('NASA Television to Air Return of Three International Space␣
↪Station Crew Members',)
```

(continues on next page)

(continued from previous page)

```
arguments received: ('NASA and Iconic Museum Honor Voyager Spacecraft 40th Anniversary
↪',)
arguments received: ('NASA's Johnson Space Center Closes Through Labor Day for␣
↪Tropical Storm Harvey',)
arguments received: ('NASA Cancels Planned Media Availabilities with Astronauts',)
arguments received: ('NASA Awards $400,000 to Top Teams at Second Phase of 3D-
↪Printing Competition',)
arguments received: ('NASA Awards Contract for Center Protective Services for Glenn␣
↪Research Center',)
arguments received: ('NASA Announces Cassini End-of-Mission Media Activities',)
1 feeds processed
```

Successs! We have skipped the two items that contain the fraud we wanted to remove from the world. Notice how we were able to *modify* the feed item: we can also use that to *change* the feed content. Normally, we would use this to fix malformed feeds, but let's have some fun instead and rename Trump to Drumpf:

```
def filter(*args, feed=None, item=None, **kwargs):
    item['title'] = item.get('title', '').replace('Trump', 'Drumpf')
```

And the result:

```
$ python3 -m feed2exec --verbose fetch --force
opening local file /home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.xml
parsing feed file:///home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.
↪xml (10355 bytes)
connecting to database at ./doc/feed2exec.db
arguments received: ('President Drumpf Welcomes Home Record-breaking NASA Astronaut␣
↪Peggy Whitson',)
arguments received: ('Three International Space Station Crewmates Safely Return to␣
↪Earth',)
arguments received: ('NASA Statement on Nomination for Agency Administrator',)
arguments received: ('NASA Television to Air Return of Three International Space␣
↪Station Crew Members',)
arguments received: ('NASA and Iconic Museum Honor Voyager Spacecraft 40th Anniversary
↪',)
arguments received: ('NASA's Johnson Space Center Closes Through Labor Day for␣
↪Tropical Storm Harvey',)
arguments received: ('NASA Cancels Planned Media Availabilities with Astronauts',)
arguments received: ('NASA Awards $400,000 to Top Teams at Second Phase of 3D-
↪Printing Competition',)
arguments received: ('NASA Awards Contract for Center Protective Services for Glenn␣
↪Research Center',)
arguments received: ('NASA Announces Cassini End-of-Mission Media Activities',)
1 feeds processed
```

I know, absolutely hilarious, right? More seriously, this is also how the *feed2exec.plugins.html2text* filter works, which is enabled by default and helps the email output plugin do its job by turning HTML into text. At this point, the only limit is your knowledge of Python programming and your imagination!

Output plugins

Output plugins are another beast entirely. While they operate with the same principle than filter plugins (search path and function signature are similar), they are designed to actually output something for each new feed item found. This can be anything: a file, email, HTTP request, whatever. If there is a commandline tool that does what you need, it is probably simpler to just call the `exec` plugin and there are numerous examples of this in the sample configuration file. For more complex things, however, it may be easier to actually write this as a Python.

Basic arguments

For our example, we'll write an archival plugin which writes each new entry to a file hierarchy. First, we start with the same simple function signature as filters, except we name it output:

```
def output(*args, feed=None, item=None, **kwargs):
    pass
```

This is the equivalent of the `null` plugin and basically outputs nothing at all. To archive the feed items, we'll need to look at the link element feedparser gives us. Let's see what that looks like for the NASA feed:

```
def output(*args, feed=None, item=None, **kwargs):
    # only operate on items that actually have a link
    if item.get('link'):
        print(item.get('link', ''))
    else:
        logging.info('no link for feed item %s, not archiving', item.get('title'))
```

Note: Note that we try to make plugins silent in general. You can use `logging.info()` to have things show up in `--verbose` and `logging.debug()` for `--debug` but by default, your plugin should be silent unless there's an error that requires the user's intervention, in which case you should use `logging.warning()` for transient errors that may be automatically recovered and `logging.error()` for errors that require user intervention. This is to allow users to ignore warnings safely.

Note that here we first check to see if the feed item actually *has* a link - not all feeds do! After adding the above to our `trumpery` plugin and adding it as an output plugin:

```
[NASA]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
output = trumpery
filter = trumpery
```

We can try to see what happens when we call it:

```
$ python3 -m feed2exec --verbose fetch --force
opening local file /home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.xml
parsing feed file:///home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.
↪xml (10355 bytes)
connecting to database at ./doc/feed2exec.db
http://www.nasa.gov/press-release/president-trump-welcomes-home-record-breaking-nasa-
↪astronaut-peggy-whitson
http://www.nasa.gov/press-release/three-international-space-station-crewmates-safely-
↪return-to-earth
http://www.nasa.gov/press-release/nasa-statement-on-nomination-for-agency-
↪administrator
http://www.nasa.gov/press-release/nasa-television-to-air-return-of-three-
↪international-space-station-crew-members
http://www.nasa.gov/press-release/nasa-and-iconic-museum-honor-voyager-spacecraft-
↪40th-anniversary
http://www.nasa.gov/press-release/nasa-s-johnson-space-center-closes-through-labor-
↪day-for-tropical-storm-harvey
http://www.nasa.gov/press-release/nasa-cancels-planned-media-availabilities-with-
↪astronauts
http://www.nasa.gov/press-release/nasa-awards-400000-to-top-teams-at-second-phase-of-
↪3d-printing-competition
```

(continues on next page)

(continued from previous page)

```
http://www.nasa.gov/press-release/nasa-awards-contract-for-center-protective-services-
↪for-glenn-research-center
http://www.nasa.gov/press-release/nasa-announces-cassini-end-of-mission-media-
↪activities
1 feeds processed
```

Sanitizing contents

Good. Those are the URLs we want to save to disk. Let's start by just writing those to a file. We will also use a simple *slug* function to make a filesystem-safe name from the feed title and save those files in a pre-determined location:

```
import logging
import os.path
from feed2exec.utils import slug

ARCHIVE_DIR='/run/user/1000/feed-archives/'

def output(*args, feed=None, item=None, **kwargs):
    # make a safe path from the item name
    path = slug(item.get('title', 'no-name'))
    # put the file in the archive directory
    path = os.path.join(ARCHIVE_DIR, path)
    # only operate on items that actually have a link
    if item.get('link'):
        # tell the user what's going on, if verbose
        # otherwise, we try to stay silent if all goes well
        logging.info('saving feed item %s to %s from %s',
                     item.get('title'), path, item.get('link'))
        # open the file
        with open(path, 'w') as archive:
            # write the response
            archive.write(item.get('link'))
    else:
        logging.info('no link for feed item %s, not archiving', item.get('title'))
```

Now I know this may look like a huge step from the previous one but I'm sorry, I couldn't find a simpler second step. :) The output now looks like this:

```
$ python3 -m feed2exec --config ./doc/ --verbose fetch --force
opening local file /home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.xml
parsing feed file:///home/anarcat/src/feed2exec/feed2exec/tests/files/breaking_news.
↪xml (10355 bytes)
connecting to database at ./doc/feed2exec.db
saving feed item President Drumpf Welcomes Home Record-breaking NASA Astronaut Peggy␣
↪Whitson to /run/user/1000/president-drumpf-welcomes-home-record-breaking-nasa-
↪astronaut-peggy-whitson from http://www.nasa.gov/press-release/president-trump-
↪welcomes-home-record-breaking-nasa-astronaut-peggy-whitson
saving feed item Three International Space Station Crewmates Safely Return to Earth␣
↪to /run/user/1000/three-international-space-station-crewmates-safely-return-to-
↪earth from http://www.nasa.gov/press-release/three-international-space-station-
↪crewmates-safely-return-to-earth
saving feed item NASA Statement on Nomination for Agency Administrator to /run/user/
↪1000/nasa-statement-on-nomination-for-agency-administrator from http://www.nasa.gov/
↪press-release/nasa-statement-on-nomination-for-agency-administrator
saving feed item NASA Television to Air Return of Three International Space Station␣
↪Crew Members to /run/user/1000/nasa-television-to-air-return-of-three-international-
↪space-station-crew-members from http://www.nasa.gov/press-release/nasa-television-
↪to-air-return-of-three-international-space-station-crew-members
```

(continues on next page)

(continued from previous page)

```
saving feed item NASA and Iconic Museum Honor Voyager Spacecraft 40th Anniversary to /
↪run/user/1000/nasa-and-iconic-museum-honor-voyager-spacecraft-40th-anniversary from␣
↪http://www.nasa.gov/press-release/nasa-and-iconic-museum-honor-voyager-spacecraft-
↪40th-anniversary
saving feed item NASA's Johnson Space Center Closes Through Labor Day for Tropical␣
↪Storm Harvey to /run/user/1000/nasa-s-johnson-space-center-closes-through-labor-day-
↪for-tropical-storm-harvey from http://www.nasa.gov/press-release/nasa-s-johnson-
↪space-center-closes-through-labor-day-for-tropical-storm-harvey
saving feed item NASA Cancels Planned Media Availabilities with Astronauts to /run/
↪user/1000/nasa-cancels-planned-media-availabilities-with-astronauts from http://www.
↪nasa.gov/press-release/nasa-cancels-planned-media-availabilities-with-astronauts
saving feed item NASA Awards $400,000 to Top Teams at Second Phase of 3D-Printing␣
↪Competition to /run/user/1000/nasa-awards-400-000-to-top-teams-at-second-phase-of-
↪3d-printing-competition from http://www.nasa.gov/press-release/nasa-awards-400000-
↪to-top-teams-at-second-phase-of-3d-printing-competition
saving feed item NASA Awards Contract for Center Protective Services for Glenn␣
↪Research Center to /run/user/1000/nasa-awards-contract-for-center-protective-
↪services-for-glenn-research-center from http://www.nasa.gov/press-release/nasa-
↪awards-contract-for-center-protective-services-for-glenn-research-center
saving feed item NASA Announces Cassini End-of-Mission Media Activities to /run/user/
↪1000/nasa-announces-cassini-end-of-mission-media-activities from http://www.nasa.
↪gov/press-release/nasa-announces-cassini-end-of-mission-media-activities
```

Sweet! Now it's not really nice to save this in `/run/user/1000`. I just chose this directory because it was a safe place to write but it's not a persistent directory. Best make that configurable, which is where plugin arguments come in.

User configuration

You see that `*args` parameter? That comes straight from the configuration file. So you could set the path in the configuration file, like this:

```
[NASA]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
output = trumpery
args = /srv/archives/nasa/
filter = trumpery
```

We also need to modify the plugin to fetch that configuration, like this:

```
def output(*args, feed=None, item=None, **kwargs):
    # make a safe path from the item name
    path = slug(item.get('title', 'no-name'))
    # take the archive dir from the user or use the default
    archive_dir = ' '.join(args) if args else DEFAULT_ARCHIVE_DIR
    # put the file in the archive directory
    path = os.path.join(archive_dir, path)
    # [...]
    # rest of the function unchanged
```

Making HTTP requests

And now obviously, we only saved the link itself, not the link *content*. For that we need some help from the `requests` module, and do something like this:

```
# fetch the URL in memory
result = requests.get(item.get('link'))
if result.status_code != requests.codes.ok:
    logging.warning('failed to fetch link %s: %s',
                    item.get('link'), result.status_code)
    # make sure we retry next time
    return False
# open the file
with open(path, 'w') as archive:
    # write the response
    archive.write(result.text)
```

This will save the actual link content (`result.text`) to the file. The important statement here is:

```
# fetch the URL in memory
result = requests.get(item.get('link'))
```

which fetches the URL in memory and checks for errors. The other change in the final plugin is simply:

```
archive.write(result.text)
```

which writes the article content instead of the link.

Plugin return values

Notice how we `return False` here: this makes the plugin system avoid adding the item to the cache, so it is retried on the next run. If the plugin returns `True` or nothing (`None`), the plugin is considered to have succeeded and the entry is added to the cache. That logic is defined in `feed2exec.feeds.parse()`.

Catchup

A final thing that is missing that is critical in all plugins is to respect the `catchup` setting. It is propagated up from the commandline or configuration all the way down to plugins, through the `feed` parameters. How you handle it varies from plugin to plugin, but the basic idea is to give feedback (when verbose) of activity when the plugin is run *but* to not actually *do* anything. In our case, we simply return success, right before we fetch the URL:

```
if feed.get('catchup'):
    return True
# fetch the URL in memory
result = requests.get(item.get('link'))
```

Notice how we still fetch the actual feed content but stop before doing any permanent operation. That is the spirit of the "catchup" operation: we not only skip "write" operation, but also any operation which could slow down the "catchup": fetching stuff over the network takes time and while it can be considered a "readonly" operation as far as the local machine is concerned, we are effectively *writing* to the network so that operation shouldn't occur.

Hopefully that should get you going with most of the plugins you are thinking of writing!

Writing tests

Writing tests is essential in ensuring that the code will stay maintainable in the future. It allows for easy refactoring and can find bugs that manual testing may not, especially when you get complete coverage (although that is no garantee either).

We'll take our *archive* plugin as an example. The first step is to edit the `tests/test/test_plugins.py` file, where other plugins are tests as well. We start by creating a function named `test_archive` so that Pytest, our test bed, will find it:

```
def test_archive(tmpdir, betamax):  # noqa
    pass
```

Notice the two arguments named `tmpdir` and `betamax`. Both of those are fixtures, a pytest concept that allows to simulate an environment. In particular, the `tmpdir` fixture, shipped with pytest, allows you to easily manage (and automatically remove) temporary directories. The `betamax` fixtures is a uses the betamax module to record then replay HTTP requests.

Then we need to do something. We need to create a feed and a feed item that we can then send into the plugin. We could also directly parse an existing feed and indeed some plugins do exactly that. But our plugin is simple and we can afford to skip full feed parsing and just synthesize what we need:

```
feed = Feed('test archive', test_sample)
item = feedparser.FeedParserDict({'link': 'http://example.com/',
                                  'title': 'example site'})
```

This creates a new feed based on the `test_sample` feed. This is necessary so that the `session` is properly re-initialized in the feed item (otherwise the `betamax` fixture will not work). Then it creates a fake feed entry simply with one link and a title. Then we can call our plugin, and verify that it saves the file as we expected. The test for the most common case looks like this:

```
def test_archive(tmpdir, betamax):  # noqa
    dest = tmpdir.join('archive')
    feed = Feed('test archive', test_sample)
    item = feedparser.FeedParserDict({'link': 'http://example.com/',
                                      'title': 'example site'})
    assert archive_plugin.output(str(dest), feed=feed, item=item)
    assert dest.join('example-site').check()
```

Then we can try to run this with `pytest-3`:

```
[1084]anarcat@curie:feed2exec$ pytest-3
============================= test session starts ==============================
platform linux -- Python 3.5.3, pytest-3.0.6, py-1.4.32, pluggy-0.4.0
rootdir: /home/anarcat/src/feed2exec, inifile: setup.cfg
plugins: profiling-1.2.11, cov-2.4.0, betamax-0.8.0
collected 26 items

feed2exec/utils.py ..
feed2exec/plugins/transmission.py .
feed2exec/tests/test_feeds.py ........
feed2exec/tests/test_main.py .....
feed2exec/tests/test_opml.py .
feed2exec/tests/test_plugins.py .........

----------- coverage: platform linux, python 3.5.3-final-0 -----------
Name                                             Stmts   Miss  Cover
--------------------------------------------------------------------
feed2exec/__init__.py                               12      0   100%
feed2exec/__main__.py                               87      1    99%
feed2exec/_version.py                                1      0   100%
feed2exec/email.py                                  81      7    91%
feed2exec/feeds.py                                 243      8    97%
```

(continues on next page)

(continued from previous page)

```
feed2exec/logging.py                              31     11    65%
feed2exec/plugins/__init__.py                     47      6    87%
feed2exec/plugins/archive.py                      23      5    78%
feed2exec/plugins/droptitle.py                     2      0   100%
feed2exec/plugins/echo.py                          8      0   100%
feed2exec/plugins/emptysummary.py                  5      0   100%
feed2exec/plugins/error.py                         2      0   100%
feed2exec/plugins/exec.py                          7      0   100%
feed2exec/plugins/html2text.py                    20      4    80%
feed2exec/plugins/ikiwiki_recentchanges.py         9      5    44%
feed2exec/plugins/maildir.py                      28      0   100%
feed2exec/plugins/mbox.py                         29      1    97%
feed2exec/plugins/null.py                          5      1    80%
feed2exec/plugins/transmission.py                 20      0   100%
feed2exec/plugins/wayback.py                      20      0   100%
feed2exec/tests/__init__.py                        0      0   100%
feed2exec/tests/conftest.py                        3      0   100%
feed2exec/tests/fixtures.py                       19      0   100%
feed2exec/tests/test_feeds.py                    124      0   100%
feed2exec/tests/test_main.py                      90      0   100%
feed2exec/tests/test_opml.py                      17      0   100%
feed2exec/tests/test_plugins.py                  162      0   100%
feed2exec/utils.py                                41     12    71%
------------------------------------------------------------------
TOTAL                                           1136     61    95%


========================== 26 passed in 10.83 seconds ==========================
```

Notice the test coverage: we only have 78% test coverage for our plugin. This means that some branches of the code were not executed at all! Let's see if we can improve that. Looking at the code, I see there are some conditionals for error handling. So let's simulate an error, and make sure that we don't create a file on error:

```
dest.remove()
item = feedparser.FeedParserDict({'link': 'http://example.com/404',
                                  'title': 'example site'})
assert not archive_plugin.output(str(dest), feed=feed, item=item)
assert not dest.join('example-site').check()
```

There. Let's see the effect on the test coverage:

```
[1085]anarcat@curie:feed2exec2$ pytest-3 feed2exec/tests/test_plugins.py::test_archive
============================== test session starts ==============================
platform linux -- Python 3.5.3, pytest-3.0.6, py-1.4.32, pluggy-0.4.0
rootdir: /home/anarcat/src/feed2exec, inifile: setup.cfg
plugins: profiling-1.2.11, cov-2.4.0, betamax-0.8.0
collected 10 items

feed2exec/tests/test_plugins.py .

----------- coverage: platform linux, python 3.5.3-final-0 -----------
Name                                          Stmts   Miss  Cover
------------------------------------------------------------------
feed2exec/__init__.py                            12      0   100%
feed2exec/__main__.py                            87     87     0%
feed2exec/_version.py                             1      0   100%
```

(continues on next page)

(continued from previous page)

```
feed2exec/email.py                                 81     64    21%
feed2exec/feeds.py                                243    172    29%
feed2exec/logging.py                               31     31     0%
feed2exec/plugins/__init__.py                      47     38    19%
feed2exec/plugins/archive.py                       23      3    87%
feed2exec/plugins/droptitle.py                      2      2     0%
feed2exec/plugins/echo.py                           8      3    62%
feed2exec/plugins/emptysummary.py                   5      5     0%
feed2exec/plugins/error.py                          2      2     0%
feed2exec/plugins/exec.py                           7      7     0%
feed2exec/plugins/html2text.py                     20     13    35%
feed2exec/plugins/ikiwiki_recentchanges.py          9      9     0%
feed2exec/plugins/maildir.py                       28     19    32%
feed2exec/plugins/mbox.py                          29     29     0%
feed2exec/plugins/null.py                           5      5     0%
feed2exec/plugins/transmission.py                  20     12    40%
feed2exec/plugins/wayback.py                       20     20     0%
feed2exec/tests/__init__.py                         0      0   100%
feed2exec/tests/conftest.py                         3      0   100%
feed2exec/tests/fixtures.py                        19      6    68%
feed2exec/tests/test_feeds.py                     124    101    19%
feed2exec/tests/test_main.py                       90     90     0%
feed2exec/tests/test_opml.py                       17     17     0%
feed2exec/tests/test_plugins.py                   166    123    26%
feed2exec/utils.py                                 41     16    61%
-------------------------------------------------------------------
TOTAL                                            1140    874    23%


============================ 1 passed in 2.46 seconds ============================
```

Much better! Only 3 lines left to cover!

Note: Notice how I explicitly provided a path to my test. This is entirely optional. You can just run `pytest-3` and it will run the whole test suite: this method is just faster. Notice also how the coverage ratio is very low: this is normal; we are testing, after all, only *one* plugin here.

The only branches left to test in the code is the other possible error ("no link in the feed") and to test the "catchup" mode. You can see this in the actual `test_plugins.py` file distributed with this documentation.

Note: If you discover a bug associated with a single feed, you can use the betamax session and the *feed2exec.feeds.Feed.parse()* function to manually parse a feed and fire your plugin. This is how email functionality is tested: see the `feed2exec.tests.test_plugins.test_email()` function for an example.

3.4.4 See also

feed2exec(1)

3.5 Support

If you have problems or question with this project, there are several options at your disposal:

- Try to troubleshoot the issue yourself
- Chat on IRC
- File bug reports

We of course welcome other contributions like documentation, translations and patches, see the *Contribution guide* guide for more information on how to contribute to the project.

3.5.1 Troubleshooting

The basic way to troubleshoot this program is to run the same command as you did when you had an error with the `--verbose` or, if that doesn't yield satisfactory results, with the `--debug` output.

Note: The debug output outputs a lot of information and may be confusing for new users.

If you suspect there is a bug specific to your environment, you can also try to see if it is reproducible within the *Test suite*. From there, you can either file a bug report or try to fix the issue yourself, see the *Contribution guide* section for more information.

Otherwise, see below for more options to get support.

3.5.2 Chat

We are often present in realtime in the `#anarcat` channel of the Freenode network. You can join the channel using a normal IRC client or using this web interface.

3.5.3 Bug reports

We want you to report bugs you find in this project. It's an important part of contributing to a project, and all bug reports will be read and replied to politely and professionally.

We are using an issue tracker to manage issues, and this is where bug reports should be sent.

Tip: A few tips on how to make good bug reports:

- Before you report a new bug, review the existing issues in the online issue tracker to make sure the bug has not already been reported elsewhere.
- The first aim of a bug report is to tell the developers exactly how to reproduce the failure, so try to reproduce the issue yourself and describe how you did that.
- If that is not possible, just try to describe what went wrong in detail. Write down the error messages, especially if they have numbers.
- Take the necessary time to write clearly and precisely. Say what you mean, and make sure it cannot be misinterpreted.
- Include the output of `--version` and `--debug` in your bug reports. See the issue template for more details about what to include in bug reports.

If you wish to read more about issues regarding communication in bug reports, you can read How to Report Bugs Effectively which takes about 30 minutes.

Warning: The output of the `--debug` may show information you may want to keep private. Do review the output before sending it in bug reports.

3.5.4 Commercial support

The project maintainers are available for commercial support for this software. If you have a feature you want to see prioritized or have a bug you absolutely need fixed, you can sponsor this development. Special licensing requirements may also be negociated if necessary. See *Contact* for more information on how to reach the maintainers.

3.6 Contribution guide

This document outlines how to contribute to this project. It details instructions on how to submit issues, bug reports and patches.

Before you participate in the community, you should agree to respect the *Code of Conduct*.

3.6.1 Positive feedback

Even if you have no changes, suggestions, documentation or bug reports to submit, even just positive feedback like "it works" goes a long way. It shows the project is being used and gives instant gratification to contributors. So we welcome emails that tell us of your positive experiences with the project or just thank you notes. Head out to contact for contact informations or submit a closed issue with your story.

You can also send your "thanks" through saythanks.io.

3.6.2 Documentation

We love documentation!

The documentation resides in various Sphinx documentations and in the README file. Those can can be edited online once you register and changes are welcome through the normal patch and merge request system.

Issues found in the documentation are also welcome, see below to file issues in our tracker.

3.6.3 Issues and bug reports

We want you to report issuess you find in the software. It is a recognized and important part of contributing to this project. All issues will be read and replied to politely and professionnally. Issues and bug reports should be filed on the issue tracker.

Issue triage

Issue triage is a useful contribution as well. You can review the issues in the project page and, for each issue:

- try to reproduce the issue, if it is not reproducible, label it with `more-info` and explain the steps taken to reproduce
- if information is missing, label it with `more-info` and request specific information
- if the feature request is not within the scope of the project or should be refused for other reasons, use the `wontfix` label and close the issue
- mark feature requests with the `enhancement` label, bugs with `bug`, duplicates with `duplicate` and so on...

Note that some of those operations are available only to project maintainers, see below for the different statuses.

Security issues

Security issues should first be disclosed privately to the project maintainers (see *Contact*), which support receiving encrypted emails through the usual OpenPGP key discovery mechanisms.

This project cannot currently afford bounties for security issues. We would still ask that you coordinate disclosure, giving the project a reasonable delay to produce a fix and prepare a release before public disclosure.

Public recognition will be given to reporters security issues if desired. We otherwise agree with the Disclosure Guidelines of the HackerOne project, at the time of writing.

3.6.4 Patches

Patches can be submitted through merge requests on the project page.

Some guidelines for patches:

- A patch should be a minimal and accurate answer to exactly one identified and agreed problem.
- A patch must compile cleanly and pass project self-tests on all target platforms.
- A patch commit message must consist of a single short (less than 50 characters) line stating a summary of the change, followed by a blank line and then a description of the problem being solved and its solution, or a reason for the change. Write more information, not less, in the commit log.
- Patches should be reviewed by at least one maintainer before being merged.

Project maintainers should merge their own patches only when they have been approved by other maintainers, unless there is no response within a reasonable timeframe (roughly one week) or there is an urgent change to be done (e.g. security or data loss issue).

As an exception to this rule, this specific document cannot be changed without the consensus of all administrators of the project.

> Note: Those guidelines were inspired by the Collective Code Construct Contract. The document was found to be a little too complex and hard to read and wasn't adopted in its entirety. See this discussion for more information.

Patch triage

You can also review existing pull requests, by cloning the contributor's repository and testing it. If the tests do not pass (either locally or in the online Continuous Integration (CI) system), if the patch is incomplete or otherwise does not respect the above guidelines, submit a review with "changes requested" with reasoning.

3.6.5 Testing

Running tests is strongly recommended before filing issues and submitting patches. Patches that break tests will not be accepted. We also aim to have complete test coverage, so you may be requested to submit a test alongside new features or bugfixes. See the *Test suite* section for more information.

3.6.6 Membership

There are three levels of membership in the project, Administrator (also known as "Owner" in GitHub or GitLab), Maintainer (also known as "Member" on GitHub or "Developer" on GitLab), or regular users (everyone with or without an account). Anyone is welcome to contribute to the project within the guidelines outlined in this document, regardless of their status, and that includes regular users.

Maintainers can:

- do everything regular users can
- review, push and merge pull requests
- edit and close issues

Administrators can:

- do everything maintainers can
- add new maintainers
- promote maintainers to administrators

Regular users can be promoted to maintainers if they contribute to the project, either by participating in issues, documentation or pull requests.

Maintainers can be promoted to administrators when they have given significant contributions for a sustained timeframe, by consensus of the current administrators. This process should be open and decided as any other issue.

3.6.7 Release process

To make a release:

1. make sure tests pass:

```
tox
```

2. generate release notes with:

```
gbp dch
```

3. tag the release according to Semantic Versioning rules:

```
git tag -s x.y.z
```

4. build and test the Python package:

```
python3 setup.py bdist_wheel
sudo pip3 install dist/*.whl
feed2exec --version
sudo pip3 uninstall feed2exec
```

5. build and test the debian package:

```
gbp buildpackage
sudo dpkg -i ../feed2exec_*.deb
feed2exec --version
sudo dpkg --remove feed2exec
```

6. push changes:

```
git push
twine upload dist/*
dput ../feed2exec*.changes
```

7. edit the tag and copy-paste the changelog entry

3.7 Code of Conduct

3.7.1 Contributor Covenant Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting one of the persons listed below. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project maintainers is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Project maintainers are encouraged to follow the spirit of the Django Code of Conduct Enforcement Manual when receiving reports.

Contacts

The following people have volunteered to be available to respond to Code of Conduct reports. They have reviewed existing literature and agree to follow the aforementioned process in good faith. They also accept OpenPGP-encrypted email:

- Antoine Beaupré anarcat@debian.org

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4.

Changes

The Code of Conduct was modified to refer to *project maintainers* instead of *project team* and small paragraph was added to refer to the Django enforcement manual.

> Note: We have so far determined that writing an explicit enforcement policy is not necessary, considering the available literature already available online and the relatively small size of the community. This may change in the future if the community grows larger.

3.8 Changelog

```
feed2exec (0.14.0) unstable; urgency=medium

  bugfix release, with a small new feature

  * warn instead of crashing on invalid dates
  * expand and cleanup date validation routines
  * use dateparser module if available
  * recommend the dateparser module for better dates handling
  * handle broken pipe correctly from plugins
  * add JSON output plugin
  * bump standards, switch to dh-compat, no change

 -- Antoine Beaupré <anarcat@debian.org>  Tue, 26 Feb 2019 17:00:07 -0500

feed2exec (0.13.0) unstable; urgency=medium

  [ Antoine Beaupré ]
  * python 3.7 support, default on python:latest now
  * wayback machine API change: HEAD not supported anymore
  * fix strange crash in wayback plugin
  * comply with new feedparser deprecation warning
  * switch to logging.warning to comply with deprecation warning
  * disable cov which crashes on sqlite3.IntegrityError
  * explain how tests are hooked up together.
  * fix typo, thanks gagz
  * include post timestamp in ikiwiki GUIDs

  [ Kang ]
  * According to RFC5064, change header 'Archive-At' to 'Archived-At'

 -- Antoine Beaupré <anarcat@debian.org>  Sat, 16 Feb 2019 10:38:09 -0500

feed2exec (0.12.0) unstable; urgency=medium

  [ Christophe Moille ]
  * feed2exec.ini no more in ~/.config/feed2exec

  [ Antoine Beaupré ]
  * synchronize commandline usage with usage docs
  * fix remaining instances of old config file path
  * add more badges incl. say thanks
  * add --catchup flag to parse as well
  * fix documentation for catchup flag
  * make the echo plugin more manifest when debugging
  * fix planet test failure on older html2text
  * add failing test for issue #5
  * fix parsing of empty links

 -- Antoine Beaupré <anarcat@debian.org>  Thu, 14 Jun 2018 12:31:38 -0400

feed2exec (0.11.0) unstable; urgency=medium

  major release: two months of bugfixes, major refactoring and
  documentation overhaul.
```

(continues on next page)

(continued from previous page)

```
  * bugfixes:
    * create missing directory in archive plugin
    * deal with feeds without a title
    * fix crash in mbox logging
    * fix crash when running without a config file
    * handle missing content-location header from the wayback machine
    * fix crash when using a relative path with --config
  * features:
    * allow shorter plugin names on the commandline: the full plugin load
      path is quite a mouthful. This makes it easier for our users and allow
      them to use only the shorter module name, relative to the
      `feed2exec.plugins` path, on the commandline only.  We keep the
      configuration file and internal use with the full plugin path. This is
      to avoid double-resolving the path and ensures the configuration file
      has a reliable plugin name.
  * major API refactoring, more details in d0b770a:
    * force API consumers to specify an explicit path instead of doing
      load-time guessing.
    * use composition instead of inheritance in the feed manager, to
      reduce ambiguity
    * locking is now in the FeedManager dispatch command
    * plugins are now ran serially even when running in parallel, although
      plugins are, as a whole, executed in parallel with the parsing, which
      is the main performance improvement we are looking for in parallelism
      anyways, because parsing is the slow part. performance tests don't
      show any significant degradation in performance
  * minor API changes:
    * allow find_test_file to return the test directory, to permit listing
      test artifacts
    * shorten and explicit variable names
    * rename test_db fixture to db_path to harmonize with conf_path fixture
    * also harmonize class names between conf and cache storage
  * extensive documentation refactoring:
    * document test writing
    * split code of conduct and contribution guide
    * fix formatting error in plugins docs and broken links in design docs
    * add security disclosure guidelines and contact
    * add support section inspired by Monkeysign
    * add issue template, integrated with GitLab
    * expand design documentation to provide a quick tour of the code
    * mention --catchup in cache purge and plugin deletion limitation
    * count lines of code without tests: it is unfair to compare out line
      count with the others because we have a much larger test suite, which
      has exploded in recent releases
    * suggest positive feedback as a contribution
    * fix syntax error and mention caveat of toot/mastodon integration
      after tests
  * test suite improvements;
    * add test suite for archive plugin
    * use pytest parametrization for opml test files
    * enable debug-level log capture in the catchlog module
    * accept OPML test files without resulting .ini file
    * refactor temp db use to simplify test suite
    * move OPML main test along with other OPML tests

 -- Antoine Beaupré <anarcat@debian.org>  Mon, 29 Jan 2018 11:12:36 -0500
```

(continues on next page)

(continued from previous page)

```
feed2exec (0.10.0) unstable; urgency=medium

  * bugfies:
    * security: avoid possible config setting override, see 2a49300 for
      details
    * follow redirections in wayback machine
    * remove useless db query when forcing
  * features
    * add transmission plugin, to write torrents to specific folders
      safely
    * add ikiwiki recentchanges filter: fixes recent changes summaries to
      fetch links properly
    * add linkchecker examples: allows users to check their publications
      for broken links
    * add shortcut parse command: allows running a single feed with a
      temporary configuration
    * make echo plugin a little more useful by *only* showing the passed
      arguments
  * documentation:
    * document catchup properly
    * add missing documentation about some add parameters in manpage
  * massive API refactoring:
    * plugins are now responsible for handling the "catchup" setting
    * FeedStorage is gone, replaced with a FeedManager
    * add a Feed object which has the parse/fetch functions and holds the
      session singleton

 -- Antoine Beaupré <anarcat@debian.org>  Sun, 05 Nov 2017 15:38:26 -0500

feed2exec (0.9.0) unstable; urgency=medium

  * API changes:
    * config file moved from ~/.config/feed2exec/feed2exec.ini to
      ~/.config/feed2exec.ini. move it by hand to keep feed2exec working
    * database cache moved from ~/.config/feed2exec/feed2exec.db to
      ~/.cache/feed2exec.db. move by hand to keep feed2exec working or
      regenerate with `feed2exec fetch --catchup`
    * use sessions everywhere and betamax for all queries
    * refactor feeds storage to remove global
    * completely rework class hierarchy
    * use requests-file instead of custom file:// handler
    * new dependencies: pyxdg and requests-file
  * bugfixes:
    * package should be arch: all, like other python packages, not any
  * features:
    * use a custom user agent

 -- Antoine Beaupré <anarcat@debian.org>  Thu, 19 Oct 2017 15:48:19 -0400

feed2exec (0.8.0) unstable; urgency=medium

  * API changes: feed2exec.feeds.fetch now returns bytes, as documented
  * bugfixes:
    * fix crash importing Liferea feed with folders, noticed by pabs
    * properly encode From headers
    * fix double-encoding issues in all output plugins
  * features:
```

(continues on next page)

(continued from previous page)

```
    * bash completion
    * add basic opml unit tests
    * opml folder support: save the parent "folder" element in config if
      relevant
    * allow duplicate feed imports by abusing the folder name
    * expanded test coverage from 89 to 93%
  * documentation:
    * expand tests documentation and add plugin design docs
    * expand on the use of vcr
    * expand the design document
    * cross-reference the two manpages

 -- Antoine Beaupré <anarcat@debian.org>  Wed, 18 Oct 2017 14:41:44 -0400

feed2exec (0.7.0) unstable; urgency=medium

  * API changes: cache skipped only if plugin returns False, not None
  * bugfixes:
    * correctly skip feeds generating fetch errors
    * add unit tests for droptitle and make it actually work
    * do not crash on empty config files
  * new plugins:
    * 'wayback' to save feed items to wayback machine
    * 'archive' to save to the local filesystem
  * documentation:
    * make build reproducible by using local doc objects
    * move design and known issues to manpage
    * add feed2exec-plugins manpage, including plugin writing
      documentation and extended plugins docs
    * fix pause and catchup descriptions
    * move documentation to RTD
    * silence docs build errors

 -- Antoine Beaupré <anarcat@debian.org>  Thu, 12 Oct 2017 16:10:02 -0400

feed2exec (0.6.0) unstable; urgency=medium

  * API-breaking changes:
    * use 'item' vocabulary consistently in API
    * allow filters to skip entry by setting the "skip" field
    * separate filter arguments (`filter_args`) from output arguments
      (`args`)
    * officially drop support for Python 2
  * features
    * add sample plugin to drop feed items matching a certain title
      (`droptitle`)
    * fix sample tweet to avoid extraneous padding
    * add transmission exec to sample config
    * do *not* wrap links even in references
    * add some limited parallelism tests
    * handle http errors more gracefully
  * bugfixes
    * html2text got a new release which broke tests, update tests and skip
      older releases
  * documentation fixes
    * clarify error message from plugin exceptions
    * expand API documentation
```

(continues on next page)

(continued from previous page)

```
    * note that feed2exec doesn't take care of IMAP folder subscriptions:
      you'll need to subscribe to new feeds by hand if you use that feature
      for now.
    * use tox in the release process, slower but more reliable
    * mark this as beta

 -- Antoine Beaupré <anarcat@debian.org>  Thu, 05 Oct 2017 14:04:16 -0400

feed2exec (0.5.1) unstable; urgency=medium

  * regenerate planet test output based on new feed
  * fix release process to workaround recent issues
  * update test suite results with feedparser 5.2.1
  * add minimal test suite documentation
  * fix typo in gbp.conf

 -- Antoine Beaupré <anarcat@debian.org>  Thu, 21 Sep 2017 18:50:54 -0400

feed2exec (0.5) unstable; urgency=medium

  * add mbox output format
  * switch to 8-bit email encodings, drop QP
  * remove useless platforms tag
  * fix tests on gitlab, no chmod allowed there
  * add fancy badges for pipeline and coverage status
  * add more generic feed test procedures
  * refactor email generation to move to its own module
  * correction: rss2email has filters
  * make sure github filter actually works
  * add example for the emptysummary filter

 -- Antoine Beaupré <anarcat@debian.org>  Thu, 21 Sep 2017 11:17:28 -0400

feed2exec (0.4) unstable; urgency=medium

  * switch to Python 3 style format strings: you need to switch from
    %(link) to {item.link}. feed parameter are also available, for example
    {feed.name} or {feed.url}. see this document for details on the syntax:
    https://docs.python.org/3/library/string.html#format-string-syntax
  * this allows more fancy formatting which gives us, for example,
    podcasting capabilities.
  * a sample config file documenting all parameters
  * add syslog support through advanced logging module from ecdysis
  * show message when done, useful for syslog
  * add sample config file
  * fix feedparser URL sanitization
  * fix issues with empty github feeds
  * note issue with SQLite locking
  * refactor test suite to regroup normalization tests
  * fix displayed path for maildir messages
  * simplify test by not running plugins twice
  * push test coverage from 87 to 90%

 -- Antoine Beaupré <anarcat@debian.org>  Thu, 14 Sep 2017 17:20:57 -0400

feed2exec (0.3) unstable; urgency=medium
```

(continues on next page)

(continued from previous page)

```
  * pause and catchup support
  * PyPI release
  * add examples to implement Twitter and Mastodon output

 -- Antoine Beaupré <anarcat@debian.org>  Tue, 12 Sep 2017 13:35:38 -0400

feed2exec (0.2) unstable; urgency=medium

  * multipart HTML support
  * improved plain text rendering
  * custom folder support
  * documentation fixes
  * expanded email headers
  * the ``output_args`` argument is renamed to ``args``
  * the ``maildir`` plugin has now a sane default, and uses the
    ``mailbox`` parameter instead of the first argument of ``output_args``
  * add ``--force`` parameter
  * make the html2text filter enabled by default in maildir

 -- Antoine Beaupré <anarcat@debian.org>  Mon, 11 Sep 2017 21:06:10 -0400

feed2exec (0.1) unstable; urgency=medium

  * first alpha release: maildir, exec support, parallelism

 -- Antoine Beaupré <anarcat@debian.org>  Mon, 11 Sep 2017 21:05:13 -0400
```

3.9 License

3.9.1 GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 November 2007

3.9.2 Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

3.9.3 TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what

server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies

made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY

AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

3.9.4 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 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 Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see http://www.gnu.org/licenses/.

3.10 Contact

This program was written by Antoine Beaupré. Please do not send email to maintainers privately unless you are looking for paid consulting or support. See *Contribution guide* for more information about how to collaborate on this project.

As a special exception, security issues can be reported privately using this contact information, where OpenPGP key material is also available.

3.10.1 Credits

This project was also inspired by similar projects:

- feed2imap
- rss2email
- feed2tweet

Special thanks also to Micah Anderson, Paul Wise and Silvio Rhatto for early testing and feedback.

Python Module Index

f

Index

R

S