
cupage Documentation

Release 0.8.2

James Rowe

Sep 27, 2017

Contents

1	Background	3
1.1	Database	3
2	Configuration	5
2.1	frequency option	6
2.2	match option	6
2.3	match_type option	6
2.4	select option	6
2.5	selector option	6
2.6	site option	7
2.7	url option	7
3	Usage	9
3.1	Options	9
3.2	Commands	9
4	cupage.py	11
4.1	Check for updates on web pages	11
4.2	SYNOPSIS	11
4.3	DESCRIPTION	11
4.4	OPTIONS	11
4.5	COMMANDS	12
4.6	CONFIGURATION FILE	13
4.7	BUGS	13
4.8	AUTHOR	13
4.9	RESOURCES	13
4.10	COPYING	13
5	Frequently Asked Questions	15
6	API documentation	17
6.1	Site	17
6.2	Command line	19
6.3	Utilities	20
7	Release HOWTO	23
7.1	Test	23

7.2	Prepare release	23
7.3	Update PyPI	23
8	Appendix	25
9	Indices and tables	27
	Python Module Index	29

cupage checks web pages and displays changes from the last run that match a given criteria. Its original purpose was to check web pages for new software releases, but it is easily configurable and can be used for other purposes.

It is written in [Python](#), and requires v2.6 or later. *cupage* is released under the [GPL v3](#)

Contents:

CHAPTER 1

Background

I had been looking for a better way to help me keep on top of software releases for the projects I'm interested in, be that either personally or for things we use at work.

Some projects have [Atom](#) feeds, some have mailing lists just for release updates, some post updates on sites like [freshmeat](#) and some have no useful update watching mechanism at all. Tracking all these resources is annoying and a simple unified solution would be much more workable.

cupage is that solution, at least for my purposes. Maybe it could be for you too!

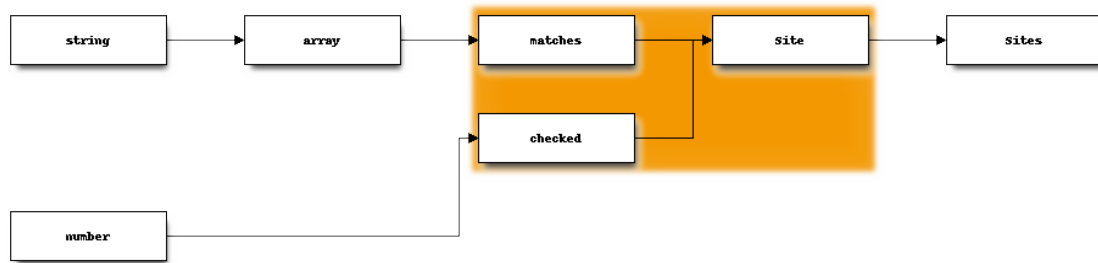
Database

With a local, unified tool we would instantly gain easy access to the updates database for use from other tools and applications.

JSON (JavaScript Object Notation) was chosen as it is simple to read and write, especially so from [Python](#) using the [json](#) module¹.

The database is a simple serialisation of the *cupage.Sites* object. The *cupage.Sites* object is a container for *cupage.Site* objects. Only persistent data from *cupage.Site* objects that can not be regenerated from the configuration file is stored in the database, namely last check time stamp and the current matches.

¹ [Pickle](#) was used in versions prior to 0.3.0. The switch was made as [Pickle](#) provided no benefits over JSON, and some significant drawbacks including the lack of support for reading it from other languages.



`matches` is an array, and contains the string matches of previous **cupage** runs.

`checked` is the offset in seconds from the Unix epoch that the site was last checked. It is normally a float, but may be null.

An example database file could be:

```
{
  "geany-plugins": {
    "matches": [
      "geany-plugins-0.17.1.tar.bz2",
      "geany-plugins-0.17.1.tar.gz",
      "geany-plugins-0.17.tar.bz2",
      "geany-plugins-0.17.tar.gz",
      "geany-plugins-0.18.tar.bz2",
      "geany-plugins-0.18.tar.gz"
    ],
    "checked": 1256677592.0
  },
  "interlude": {
    "matches": [
      "interlude-1.0.tar.gz"
    ],
    "checked": null
  }
}
```


CHAPTER 2

Configuration

cupage stores its configuration in `~/.cupage.conf` by default, although you can specify a different location with the `cupage list -f` command line option.

The configuration file is a INI format file, with a section for each site definition. The section header is the site's name which will be displayed in the update output, or used to select individual sites to check on the command line. Each section consists of a section of `name=value` option pairs.

An example configuration file is below:

```
[pep8]
site = pypi
match_type = tar
[pydelicious]
site = google code
match_type = zip
[pyisbn]
url = http://www.jnrowe.ukfsn.org/_downloads/
select = pre > a
match_type = tar
frequency = 6m
[upoints]
url = http://www.jnrowe.ukfsn.org/_downloads/
select = pre > a
match_type = tar
[fruity]
site = vim-script
script = 1871
[cupage]
site = github
user = JNRowe
frequency = 1m
```

Site definitions can either be specified entirely manually, or possibly with the built-in site matchers(see [site option](#) for available options).

frequency option

The `frequency` option allows you to set a minimum time between checks for specific sites within the configuration file.

The format is `<value> <units>` where `value` can be a integer or float, and `units` must be one of the entries from the table below:

Unit	Purpose
h	Hours
d	Days
w	Week
m	Month, which is defined as 28 days
y	Year, which is defined as 13 m units

match option

If `match_type` is `re` then `match` must be a valid regular expression that will be used to match within the selected elements. For most common uses a prebuilt `match_type` already exists(see [match_type option](#)), and `re` should really only be used as a last resort.

The Python `re` module is used, and any functionality allowed by the module is available in the `match` option(with the notable exception of the `verbose` syntax).

match_type option

The `match_type` value, if used, must be one of the following:

Match type	Purpose
gem	to match rubygems archives.
re	to define custom regular expressions
tar	to match gzip/bzip2/xz compressed tar archives(default)
zip	to match zip archives

The `match_type` values simply select a predefined regular expression to use. The base match is `<name>-[\\d\\.] +([_ -](pre|rc)[\\d]+)?\\.<type>`, where `<name>` is the section name and `<type>` is the value of `match_type` for this section.

select option

The `select` option, if used, must be a valid CSS (Cascading Style Sheets) or XPath selector depending on the value of `selector` (see [selector option](#)). Unless specified CSS CASCADING STYLE SHEETS) is the default selector type.

selector option

The `selector` option, if used, must be one of the following:

Selector	Purpose
css	To select elements within the page using CSS selectors (default)
xpath	To select elements within the page using XPath selectors

site option

The `site` option, if used, must be one of the following, hopefully self-explanatory values:

Site	Added	Required options
<code>cpan</code>	<code>v0.4.0</code>	
<code>debian</code>	<code>v0.3.0</code>	
<code>failpad</code>	<code>v0.5.0</code>	
<code>github</code>	<code>v0.3.1</code>	<code>user</code> (GitHub user name)
<code>google code</code>	<code>v0.1.0</code>	
<code>hackage</code>	<code>v0.1.0</code>	
<code>pypi</code>	<code>v0.1.0</code>	
<code>vim-script</code>	<code>v0.3.0</code>	<code>script</code> (script id on the vim website)

`site` options are simply shortcuts that are provided to reduce duplication in the configuration file. They define the values necessary to check for updates on the given site.

url option

The `url` value is the location of the page to be checked for updates. If used, it must be a valid FTP (File Transfer Protocol)/HTTP (HyperText Transfer Protocol)/HTTPS (HyperText Transfer Protocol) address.

The **cupage** is run from the command prompt, and displays updates on `stdout`.

Options

- version**
show program's version number and exit
- h, --help**
show this help message and exit
- v, --verbose**
produce verbose output
- q, --quiet**
output only matches and errors

Commands

add - add definition to config file

- f <file>, --config <file>**
configuration file to read
- s <site>, --site <site>**
site helper to use
- u <url>, --url <url>**
site url to check
- t <type>, --match-type <type>**
pre-defined regular expression to use

-m <regex>, **--match** <regex>
regular expression to use with **-match-type=re**

-q <frequency>, **--frequency** <frequency>
update check frequency

-x <selector>, **--select** <selector>
content selector

--selector <type>
selector method to use

check - check sites for updates

-f <file>, **--config** <file>
configuration file to read

-d <file>, **--database** <file>
database to store page data to. Default based on **--config** value, for example **--config my_conf** will result in a default setting of **--database my_conf.db**.

See [Database](#) for details of the database format.

-c <dir>, **--cache** <dir>
directory to store page cache

This can, and in fact *should* be, shared between all cupage uses.

--no-write
don't update cache or database

--force
ignore frequency checks

-t <n>, **--timeout**=<n>
timeout for network operations

list - list definitions from config file

-f <file>, **--config** <file>
configuration file to read

-m <regex>, **--match** <regex>
match sites using regular expression

list-sites - list supported site values

remove - remove site from config

-f <file>, **--config** <file>
configuration file to read

Check for updates on web pages

Author James Rowe <jnrowe@gmail.com>

Date 2010-01-23

Copyright GPL v3

Manual section 1

Manual group Networking

SYNOPSIS

cupage.py [option]... <command>

DESCRIPTION

cupage checks web pages and displays changes from the last run that match a given criteria. Its original purpose was to check web pages for new software releases, but it is easily configurable and can be used for other purposes.

OPTIONS

--version	show program's version number and exit
-h, --help	show this help message and exit
-v, --verbose	produce verbose output
-q, --quiet	output only matches and errors

COMMANDS

add

Add definition to config file

- f <file>, --config <file>** configuration file to read
- s <site>, --site <site>** site helper to use
- u <url>, --url <url>** site url to check
- t <type>, --match-type <type>** pre-defined regular expression to use
- m <regex>, --match <regex>** regular expression to use with `--match-type=re`
- q <frequency>, --frequency <frequency>** update check frequency
- x <selector>, --select <selector>** content selector
- selector <type>** selector method to use

check

Check sites for updates

- f <file>, --config <file>** configuration file to read
- d <file>, --database <file>** database to store page data to. Default based on `cupage check -f` value, for example `--config my_conf` will result in a default setting of `--database my_conf.db`.

See [Database](#) for details of the database format.
- c <dir>, --cache <dir>** directory to store page cache

This can, and in fact *should* be, shared between all cupage uses.
- no-write** don't update cache or database
- force** ignore frequency checks
- t <n>, --timeout=<n>** timeout for network operations

list

List definitions from config file

- f <file>, --config <file>** configuration file to read
- m <regex>, --match <regex>** match sites using regular expression

list-sites

List supported site values

remove

Remove site from config

-f <file>, **--config <file>** configuration file to read

CONFIGURATION FILE

The configuration file, by default `~/.cupage.conf`, is a simple **INI** format file, with sections defining sites to check. For example:

```
[spill]
url = http://www.rpcurnow.force9.co.uk/spill/index.html
select = p a
[rails]
site = vim-script
script = 1567
```

With the above configuration file the site named **spill** will be checked at <http://www.rpcurnow.force9.co.uk/spill/index.html>, and elements matching the CSS selector **p a** will be scanned for tarballs. The site named **rails** will be checked using the **vim-script** site matcher, which requires only a **script** value to check for updates in the scripts section of <http://www.vim.org>.

Various site matchers are available, see the output of `cupage.py --list-sites`.

BUGS

None known.

AUTHOR

Written by James Rowe

RESOURCES

Home page: <http://github.com/JNRowe/cupage>

COPYING

Copyright © 2009-2013 James Rowe.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

CHAPTER 5

Frequently Asked Questions

Ask them, and perhaps they'll become frequent enough to be added here ;)

Note: The documentation in this section is aimed at people wishing to contribute to *cupage*, and can be skipped if you are simply using the tool from the command line.

Site

Note: The documentation in this section is aimed at people wishing to contribute to *cupage*, and can be skipped if you are simply using the tool from the command line.

`cupage.SITES = {}`

Site specific configuration data

`cupage.USER_AGENT = 'cupage/0.8.2 +https://github.com/JNRowe/cupage/'`

User agent to use for HTTP requests

`class cupage.Site(name, url, match_func='default', options=None, frequency=None, robots=True, checked=None, matches=None)`

Simple object for representing a web site.

`check(cache=None, timeout=None, force=False, no_write=False)`

Check site for updates.

Parameters

- **cache** (*str*) – `httplib2.Http` cache location
- **timeout** (*int*) – Timeout value for `httplib2.Http`
- **force** (*bool*) – Ignore configured check frequency
- **no_write** (*bool*) – Do not write to cache, useful for testing

find_default_matches (*content*, *charset*)

Extract matches from content.

Parameters

- **content** (*str*) – Content to search
- **charset** (*str*) – Character set for content

find_github_matches (*content*, *charset*)

Extract matches from GitHub content.

Parameters

- **content** (*str*) – Content to search
- **charset** (*str*) – Character set for content

find_hackage_matches (*content*, *charset*)

Extract matches from hackage content.

Parameters

- **content** (*str*) – Content to search
- **charset** (*str*) – Character set for content

find_sourceforge_matches (*content*, *charset*)

Extract matches from sourceforge content.

Parameters

- **content** (*str*) – Content to search
- **charset** (*str*) – Character set for content

static package_re (*name*, *ext*, *verbose=False*)

Generate a compiled `re` for the package.

Parameters

- **name** (*str*) – File name to check for
- **ext** (*str*) – File extension to check
- **verbose** (*bool*) – Whether to enable `re.VERBOSE`

static parse (*name*, *options*, *data*)

Parse data generated by `Sites.loader`.

Parameters

- **name** (*str*) – Site name from config file
- **options** (*dict*) – Site options from config file
- **data** – Stored data from database file

state

Return `Site` state for database storage.

class `cupage.Sites`

Site bundle wrapper.

load (*config_file*, *database=None*)

Read sites from a user's config file and database.

Parameters

- **config_file** (*str*) – Config file to read
- **database** (*str*) – Database file to read

save (*database*)

Save Sites to the user's database.

Parameters **database** (*str*) – Database file to write

Examples

Reading stored configuration

```
>>> sites = Sites()
>>> sites.load('support/cupage.conf', 'support/cupage.db')
>>> sites[0].frequency
360000
```

Writing updates

```
>>> sites.save('support/cupage.db')
```

Command line

Note: The documentation in this section is aimed at people wishing to contribute to *cupage*, and can be skipped if you are simply using the tool from the command line.

`cupage.cmdline.USAGE = '%(prog)s checks web pages and displays changes from the last run that match\na given criteria.'`

Command line help string, for use with `argparse`

`cupage.cmdline.main()`

Main script handler.

`cupage.cmdline.add(verbose, config, site, url, match_type, match, frequency, select, selector, name)`

`cupage.cmdline.check(verbose, config, database, cache, no_write, force, timeout, pages)`

`cupage.cmdline.list_conf(verbose, config, database, match, pages)`

`cupage.cmdline.list_sites(verbose)`

`cupage.cmdline.remove(verbose, config, pages)`

Examples

Parse command line options

```
>>> options, args = process_command_line()
```

Utilities

Note: The documentation in this section is aimed at people wishing to contribute to *cupage*, and can be skipped if you are simply using the tool from the command line.

class `cupage.utils.CupageEncoder` (*skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, encoding='utf-8', default=None*)

Custom JSON encoding for supporting datetime objects.

default (*obj*)

Handle datetime objects when encoding as JSON.

This simply falls through to `default()` if *obj* has no `isoformat` method.

Parameters *obj* – Object to encode

`cupage.utils.json_to_datetime` (*obj*)

Parse checked datetimes from cupage databases.

See `json.JSONDecoder`

Parameters *obj* – Object to decode

`cupage.utils.parse_timedelta` (*delta*)

Parse human readable frequency.

Parameters *delta* (*str*) – Frequency to parse

`cupage.utils.sort_packages` (*packages*)

Order package list according to version number.

Parameters *packages* (*list*) – Packages to sort

`cupage.utils.robots_test` (*http, url, name, user_agent='*'*)

Check whether a given URL is blocked by `robots.txt`.

Parameters

- **http** – `httplib2.Http` object to use for requests
- **url** (*str*) – URL to check
- **name** – Site name being checked
- **user_agent** (*str*) – User agent to check in `robots.txt`

The following three functions are defined for purely cosmetic reasons, as they make the calling points easier to read.

`cupage.utils.success` (*text*)

Format a success message with colour, if possible.

Parameters *text* (*str*) – Text to format

`cupage.utils.fail` (*text*)

Format a failure message with colour, if possible.

Parameters *text* (*str*) – Text to format

`cupage.utils.warn` (*text*)

Format a warning message with colour, if possible.

Parameters *text* (*str*) – Text to format

Examples

Output formatting

```
>>> success('well done!')
u'\x1b[38;5;10mwell done!\x1b[m\x1b(B'
>>> fail('unlucky!')
u'\x1b[38;5;9munlucky!\x1b[m\x1b(B'
```


Test

In the general case tests can be run via `nose2`:

```
$ nose2 -vv tests
```

When preparing a release it is important to check that *cupage* works with all currently supported Python versions, and that the documentation is correct.

Prepare release

With the tests passing, perform the following steps

- Update the version data in `cupage/_version.py`
- Update `NEWS.rst`, if there are any user visible changes
- Commit the release notes and version changes
- Create a signed tag for the release
- Push the changes, including the new tag, to the GitHub repository

Update PyPI

Create and upload the new release tarballs to PyPI:

```
$ ./setup.py sdist --formats=bztar,gztar register upload --sign
```

Fetch the uploaded tarballs, and check for errors.

You should also perform test installations from PyPI, to check the experience *cupage* users will have.

CHAPTER 8

Appendix

```
class httpplib2.Http
    Instance of Http from httpplib2
```


CHAPTER 9

Indices and tables

- `genindex`
- `modindex`
- `search`

C

cupage, ??

Symbols

- force
 - cupage-check command line option, 10
- no-write
 - cupage-check command line option, 10
- selector <type>
 - cupage-add command line option, 10
- version
 - cupage command line option, 9
- c <dir>, -cache <dir>
 - cupage-check command line option, 10
- d <file>, -database <file>
 - cupage-check command line option, 10
- f <file>, -config <file>
 - cupage-add command line option, 9
 - cupage-check command line option, 10
 - cupage-list command line option, 10
 - cupage-remove command line option, 10
- h, -help
 - cupage command line option, 9
- m <regex>, -match <regex>
 - cupage-add command line option, 9
 - cupage-list command line option, 10
- q <frequency>, -frequency <frequency>
 - cupage-add command line option, 10
- q, -quiet
 - cupage command line option, 9
- s <site>, -site <site>
 - cupage-add command line option, 9
- t <n>, -timeout=<n>
 - cupage-check command line option, 10
- t <type>, -match-type <type>
 - cupage-add command line option, 9
- u <url>, -url <url>
 - cupage-add command line option, 9
- v, -verbose
 - cupage command line option, 9
- x <selector>, -select <selector>
 - cupage-add command line option, 10

A

add() (in module cupage.cmdline), 19

C

check() (cupage.Site method), 17

check() (in module cupage.cmdline), 19

cupage (module), 1, 15

cupage command line option

- version, 9
- h, -help, 9
- q, -quiet, 9
- v, -verbose, 9

cupage-add command line option

- selector <type>, 10
- f <file>, -config <file>, 9
- m <regex>, -match <regex>, 9
- q <frequency>, -frequency <frequency>, 10
- s <site>, -site <site>, 9
- t <type>, -match-type <type>, 9
- u <url>, -url <url>, 9
- x <selector>, -select <selector>, 10

cupage-check command line option

- force, 10
- no-write, 10
- c <dir>, -cache <dir>, 10
- d <file>, -database <file>, 10
- f <file>, -config <file>, 10
- t <n>, -timeout=<n>, 10

cupage-list command line option

- f <file>, -config <file>, 10
- m <regex>, -match <regex>, 10

cupage-remove command line option

- f <file>, -config <file>, 10

CupageEncoder (class in cupage.utils), 20

D

default() (cupage.utils.CupageEncoder method), 20

F

`fail()` (in module `cupage.utils`), 20
`find_default_matches()` (`cupage.Site` method), 17
`find_github_matches()` (`cupage.Site` method), 18
`find_hackage_matches()` (`cupage.Site` method), 18
`find_sourceforge_matches()` (`cupage.Site` method), 18

H

`httplib2.Http` (built-in class), 25

J

`json_to_datetime()` (in module `cupage.utils`), 20

L

`list_conf()` (in module `cupage.cmdline`), 19
`list_sites()` (in module `cupage.cmdline`), 19
`load()` (`cupage.Sites` method), 18

M

`main()` (in module `cupage.cmdline`), 19

P

`package_re()` (`cupage.Site` static method), 18
`parse()` (`cupage.Site` static method), 18
`parse_timedelta()` (in module `cupage.utils`), 20

R

`remove()` (in module `cupage.cmdline`), 19
`robots_test()` (in module `cupage.utils`), 20

S

`save()` (`cupage.Sites` method), 19
`Site` (class in `cupage`), 17
`Sites` (class in `cupage`), 18
`SITES` (in module `cupage`), 17
`sort_packages()` (in module `cupage.utils`), 20
`state` (`cupage.Site` attribute), 18
`success()` (in module `cupage.utils`), 20

U

`USAGE` (in module `cupage.cmdline`), 19
`USER_AGENT` (in module `cupage`), 17

W

`warn()` (in module `cupage.utils`), 20