Liveweb Proxy Documentation

Release 2.0-dev

Internet Archive

August 01, 2013

CONTENTS

1 Installation **3**

2 Running liveweb-proxy **5**

3 Advanced Usage **7**

4 Documentation **9**

4.1 Development Setup 9

4.2 Liveweb Proxy Configuration 10

4.3 Error Codes 12

Liveweb proxy is a component of Internet Archive's wayback machine project.

The liveweb proxy captures the content of a web page in real time, archives it into a ARC or WARC file and returns the ARC/WARC record back to the wayback machine to process. The recorded ARC/WARC file becomes part of the wayback machine in due course of time.

Note: The liveweb project is under active development, so this documentation may not be up-to-date.

CHAPTER

ONE

INSTALLATION

Liveweb proxy can be installed using pip:

```
$ pip install liveweb
```

or, with easy_install

```
$ easy_install liveweb
```

See Development Setup if you want to work with source code.

CHAPTER

TWO

RUNNING LIVEWEB-PROXY

Liveweb proxy can be run using:

```
$ liveweb-proxy
```

To start liveweb-proxy on a different port:

```
$ liveweb-proxy -p 8080
```

To load settings from a config file:

```
$ liveweb-proxy -c liveweb.ini
```

To see the available command-line options:

```
$ liveweb-proxy --help
```

See *Configuration* section for the available config settings and command line options.

CHAPTER

THREE

ADVANCED USAGE

Under the hood, liveweb proxy used uwsgi as the http server.

If you want to tweak uwsgi parameters, you can start liveweb as:

```
$ uwsgi --master --single-interpreter --lazy --wsgi liveweb.main --processes 1 --threads 10 --http lo
```

The values of `--processes`, `--threads` and `--http` options can be changed as needed and more options can be added too.

You may have to specify *-H virtualenv_home* if you using a virtualenv.

CHAPTER

FOUR

DOCUMENTATION

4.1 Development Setup

4.1.1 Setting up

Start with getting the source code from github.

```
$ git clone git://github.com/internetarchive/liveweb.git
$ cd liveweb
```

Setup a virtualenv.

```
$ make venv
```

This will create the virtualenv in the current directory. Edit the `Makefile` if you want to setup virtualenv elsewhere.

4.1.2 Running the application

Run the application using:

```
$ make run
```

This will start the liveweb proxy at `localhost:7070`.

4.1.3 Testing using curl

Assuming the liveweb proxy is running on *localhost:7070*:

```
$ curl -s -x localhost:7070 http://httpbin.org/get | zcat
http://httpbin.org/get 204.236.238.79 20120427110218 application/json 451
HTTP/1.1 200 OK
Content-Type: application/json
Date: Fri, 27 Apr 2012 11:02:18 GMT
Server: gunicorn/0.13.4
Content-Length: 298
Connection: Close

{
  "url": "http://httpbin.org/get",
  "headers": {
    "Content-Length": "",
```

```
    "Accept-Encoding": "identity",
    "Connection": "keep-alive",
    "User-Agent": "ia_archiver(OS-Wayback)",
    "Host": "httpbin.org",
    "Content-Type": ""
  },
  "args": {},
  "origin": "207.241.237.193"
}
```

4.1.4 Running in http-passthough mode

Enable http-passthrough mode by adding the following to the config file.

```
http_passthough: true
```

Make sure caching is disabled. The http-passthough mode doesn't work with caching.

Run the application and change the browser setting to use application address (localhost:7070 by default) as http proxy.

4.1.5 Performance Testing

Test performance using Apache-Bench:

```
$ ab -X localhost:7070 -c 10 -n 100 http://www.archive.org/
```

The `-X` options is to specify the proxy server.

4.2 Liveweb Proxy Configuration

The `liveweb-proxy` can be configured using various command-line options and/or a config file.

Config file can be specified as:

```
$ liveweb-proxy -c liveweb.ini
```

or:

```
$ liveweb-proxy --config liveweb.ini
```

This section describes the available config settings. For each config setting, there is a command line option with the same name.

For example, config setting `archive-format` is available as command line argument *–archive-format*.

The config file is specified in INI format. Here is a sample config file.

```
[liveweb]

archive-format = arc

output-directory = /tmp/records

dns-timeout = 2s
```

4.2.1 Archive Settings

archive-format

Specifies the archive format. Should be one if `arc` or `warc`.

The default value is `arc`.

> **Warning:** As of now only `arc` is supported.

output-directory

Output directory to write ARC/WRC files. Default value is "records".

filename-pattern

The pattern of the filename specified as Python string formatting template. The default value is `live-%(timestamp)s-%(serial)05d.arc.gz`.

Available substitutions are `timestamp`, `serial`, `pid`, `fqdn` (fully qualified domain name) and `port`.

filesize-limit

The limit on the size of file. If a file crosses this size, it will be closed a new file will be created to write new records.

num-writers

The number of concurrent writers.

The default value is `1`.

4.2.2 Cache Settings

cache

Type of cache to use. Available options are `redis`, `sqlite` and `none`.

The default value is `none`.

redis-host

redis-port

redis-db

Redis host, port and db number. Used only when `cache=redis`.

redis-expire-time

Expire time to set in redis. Used only when `cache=redis`.

The default value is `1h` (1 hour).

redis-max-record-size

Maximum allowed size of a record that can be cached. Used only when `cache=redis`.

The default value is `100KB`.

sqlite-db

Path to the sqlite database to use. This option is valid only when `cache=sqlite`.

The default value is `liveweb.db`.

4.2.3 Timeouts and Resource Limits

default-timeout

This is the default timeout value for `connect-timeout`, `initial-data-timeout` and `read-timeout`.

The default value is `10s`.

dns-timeout

Specifies the max amount of time can a DNS resolution can take.

Python doesn't support a way to specify DNS timeout. On Linux, the dns timeout can be specified via the `RES_OPTIONS` environment variable. This enviroment variable is set at the startup of the application based on this config setting.

If unspecified, the DNS timeout is decided by the system default behavior.

See resolv.conf man page for more details.

connect-timeout

Specifies the connect timeout in seconds. Connections that take longer to establish will be aborted.

initial-data-timeout

Specifies the maximum time allowed before receiving initial data (HTTP headers) from the remote server.

read-timeout

Specifies the read timeout in seconds. This indicates the idle time. If no data is received for more than this time, the request will fail.

max-request-time

Specifies the total amout of time a HTTP request can take. If it takes more than this, the current request will fail.

The default value is `2m`.

max-response-size

Specifies the maximum allowed size of response.

The default value is `100MB`.

4.2.4 Other Settings

user-agent

Specifies the value of the `User-Agent` request header.

The default value is `ia_archiver(OS-Wayback)`.

http-passthrough

This is a boolean parameter, setting it to `true` will make it work like a http proxy with archiving. Useful for testing and recording personal browsing.

4.3 Error Codes

The application writes the errors with following codes when something fails when trying to fetch the given URL.

4.3.1 1X - Bad Input

E10 - Invalid URL

The given URL is invalid. For example:

```
http://example.com:bad-port/
```

4.3.2 2X - DNS errors

E20 - Invalid Domain

The URL has non existant domain.

E21 - DNS Timeout

The hostname couldn't be resolved within *config_dns_timeout* seconds.

4.3.3 3X - Connection Errors

E30 - Connection Refused

Connection refused by the server.

E31 - Connect Timeout

Connection couldn't be established within *config_connect_timeout* seconds.

E32 - Initial Data Timeout

Initial data (HTTP headers) couldn't be obtained within *config_initial_data_timeout* seconds.

E33 - Read Timeout

When reading data from the remote server, no data was received for *config_read_timeout* seconds.

E34 - Connection Dropped

The remote server dropped the connection before all the data was received.

E39 - Unexpected Connection Error

Unexpected connection error when receiving data from the remote server.

4.3.4 4X - Resource Limits

E40 - Response Too Big

The response length is bigger than *config_max_response_size* bytes.

E41 - Request Took Too Long

The request was not completed within *config_max_request_time* seconds.

In all these cases, the application responds back with status `200 OK` with a record contain status `302 Bad Gateway`.