Pacifica Proxy Documentation

David Brown

May 18, 2020

Contents:

1 Installation **3**
1.1 Installation in Virtual Environment . . . 3

2 Configuration **5**
2.1 CherryPy Configuration File . . . 5
2.2 Service Configuration File . . . 5
2.3 Starting the Service . . . 6

3 Example Usage **9**
3.1 Files Access . . . 9

4 Proxy Python Module **11**
4.1 Configuration Python Module . . . 11
4.2 Files Python Module . . . 11
4.3 Globals Python Module . . . 11
4.4 REST Python Module . . . 11
4.5 WSGI Python Module . . . 11

5 Indices and tables **13**

The Pacifica Proxy service provides access to internal services through internal metadata that users would know.

CHAPTER 1

Installation

The Pacifica software is available through PyPi so creating a virtual environment to install is what is shown below. Please keep in mind compatibility with the Pacifica Core services.

1.1 Installation in Virtual Environment

These installation instructions are intended to work on both Windows, Linux, and Mac platforms. Please keep that in mind when following the instructions.

Please install the appropriate tested version of Python for maximum chance of success.

1.1.1 Linux and Mac Installation

```
mkdir ~/.virtualenvs
python -m virtualenv ~/.virtualenvs/pacifica
. ~/.virtualenvs/pacifica/bin/activate
pip install pacifica-proxy
```

1.1.2 Windows Installation

This is done using PowerShell. Please do not use Batch Command.

```
mkdir "$Env:LOCALAPPDATA\virtualenvs"
python.exe -m virtualenv "$Env:LOCALAPPDATA\virtualenvs\pacifica"
& "$Env:LOCALAPPDATA\virtualenvs\pacifica\Scripts\activate.ps1"
pip install pacifica-proxy
```

CHAPTER 2

Configuration

The Pacifica Core services require two configuration files. The REST API utilizes CherryPy and review of their configuration documentation is recommended. The service configuration file is a INI formatted file containing configuration for database connections.

2.1 CherryPy Configuration File

An example of Proxy server CherryPy configuration:

```
[global]
log.screen: True
log.access_file: 'access.log'
log.error_file: 'error.log'
server.socket_host: '0.0.0.0'
server.socket_port: 8051

[/]
request.dispatch: cherrypy.dispatch.MethodDispatcher()
tools.response_headers.on: True
tools.response_headers.headers: [('Content-Type', 'application/json')]
```

2.2 Service Configuration File

The service configuration is an INI file and an example is as follows:

```
[database]
; This section contains database connection configuration

; peewee_url is defined as the URL PeeWee can consume.
; http://docs.peewee-orm.com/en/latest/peewee/database.html#connecting-using-a-
↪database-url
```

(continues on next page)

(continued from previous page)

```
peewee_url = sqliteext:///db.sqlite3

; connect_attempts are the number of times the service will attempt to
; connect to the database if unavailable.
connect_attempts = 10

; connect_wait are the number of seconds the service will wait between
; connection attempts until a successful connection to the database.
connect_wait = 20
```

2.3 Starting the Service

Starting the Proxy service can be done by two methods. However, understanding the requirements and how they apply to REST services is important to address as well. Using the internal CherryPy server to start the service is recommended for Windows platforms. For Linux/Mac platforms it is recommended to deploy the service with uWSGI.

2.3.1 Deployment Considerations

The Proxy service utilizes both connectivity to the Metadata service and the Archive Interface. As a result close connectivity to those services is a must.

2.3.2 CherryPy Server

To make running the Proxy service using the CherryPy's builtin server easier we have a command line entry point.

```
$ pacifica-proxy --help
usage: pacifica-proxy [-h] [-c CONFIG] [-p PORT] [-a ADDRESS]

Run the proxy server.

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        cherrypy config file
  -p PORT, --port PORT  port to listen on
  -a ADDRESS, --address ADDRESS
                        address to listen on
$ pacifica-proxy
[09/Jan/2019:09:17:26] ENGINE Listening for SIGTERM.
[09/Jan/2019:09:17:26] ENGINE Bus STARTING
[09/Jan/2019:09:17:26] ENGINE Set handler for console events.
[09/Jan/2019:09:17:26] ENGINE Started monitor thread 'Autoreloader'.
[09/Jan/2019:09:17:26] ENGINE Serving on http://0.0.0.0:8180
[09/Jan/2019:09:17:26] ENGINE Bus STARTED
```

2.3.3 uWSGI Server

To make running the Proxy service using uWSGI easier we have a module to be included as part of the uWSGI configuration. uWSGI is very configurable and can use this module many different ways. Please consult the uWSGI Configuration documentation for more complicated deployments.

```
$ pip install uwsgi
$ uwsgi --http-socket :8180 --master --module pacifica.proxy.wsgi
```

2.3.4 NGINX Acceleration

The Proxy service supports NGINX Acceleration, an example NGINX Configuration is used in the Travis CI pipeline. The setup is to have the Proxy service tell NGINX to redirect the request to the Archiveinterface internally. This configuration bypasses transferring the data through the Proxy service. This configuration requires the Archiveinterface and Proxy services to be accessible through the same border NGINX endpoint exposed to users.

CHAPTER 3

Example Usage

Different paths in the Proxy service have different usages. The files API is currently only provided but future paths could be added.

3.1 Files Access

The archive interface service is intended to be used by internal services to access files off the archive by file ID only. This can be easily iterated over by external users and should not be exposed externally. This service accepts a hashsum provided by the user and looks up a file ID based on that hashsum. The service then redirects the request without knowledge of the user to the archive interface to pull the file.

3.1.1 File Access API

Example curl command

```
curl http://localhost:8180/files/sha1/f90a581a5099079a8f1f582dd3643b6e060cc551
```

If the file exists the file is given as an octet-stream to the user. The disposition header is also set with the filename defined in the metadata for that file.

If the file does not exist a `404 Not Found` return code is given.

CHAPTER 4

Proxy Python Module

4.1 Configuration Python Module

4.2 Files Python Module

4.3 Globals Python Module

4.4 REST Python Module

4.5 WSGI Python Module

CHAPTER 5

Indices and tables

- genindex
- modindex
- search