Pacifica Cartd Documentation

David Brown

Dec 18, 2019

Contents:

The Pacifica Cartd service provides data staging and bundling for user consumption of data.

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-cartd
```

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-cartd
```

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 Cartd 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: 8081

[/]
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:

```
[cartd]
; This section describes cartd specific configuration

; Local directory to stage data
volume_path = /tmp/
```

(continues on next page)

(continued from previous page)

```
; Least recently used buffer time
lru_buffer_time = 0

; Bundle backend task enable/disable
bundle_task = True

[archiveinterface]
; This section describe where the archive interface is

; URL to the archive interface
url = http://127.0.0.1:8080/

[celery]
; This section describe celery task configuration

; Broker message url
broker_url = pyamqp://

; Backend task channel
backend_url = rpc://

[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
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 Cartd 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 Cartd service stages data for consumption by data users. This service (like Ingest) should be put on the edge of your infrastructure to allow for fast access. Other considerations about data transfers over these networks should also be considerred. ESNet has some good documentation on how to optimize Linux for fast data transfers.

2.3.2 CherryPy Server

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

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

Run the cart server.

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        cart config file
  --cpconfig CONFIG     cherrypy config file
  -p PORT, --port PORT  port to listen on
  -a ADDRESS, --address ADDRESS
                        address to listen on
$ pacifica-cartd-cmd dbsync
$ pacifica-cartd
[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://0.0.0.0:8081
[09/Jan/2019:09:17:26] ENGINE Bus STARTED
```

2.3.3 uWSGI Server

To make running the Cartd 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 :8081 --master --module pacifica.cartd.wsgi
```

CHAPTER 3

Example Usage

Every cart has a unique ID associated with it. For the examples following we used a uuid generated by standard Linux utilities.

```
MY_CART_UUID=`uuidgen`
```

3.1 The REST API

The REST API is available for users of the system and is in general a method based endpoint with JSON objects for data.

3.1.1 Create a Cart

Post a file to create a new cart.

Contents of file (foo.json).

id = the id being used on the Archive

path = internal structure of bundle for file placement

hashtype = hashlib hashtype used to generate hashsum

hashsum = the hash (hex value) of the file using the hashtype listed

```
{
  "fileids": [
    {"id":"foo.txt", "path":"1/2/3/foo.txt", "hashtype":"md5", "hashsum":""},
    {"id":"bar.csv", "path":"1/2/3/bar.csv", "hashtype":"md5", "hashsum":""},
    {"id":"baz.ini", "path":"2/3/4/baz.ini", "hashtype":"md5", "hashsum":""}
  ]
}
```

Post the file to the following URL.

```
curl -X POST --upload-file /tmp/foo.json http://127.0.0.1:8081/$MY_CART_UUID
```

3.1.2 Status a Cart

Head on the cart to find whether its created and ready for download.

```
curl -I -X HEAD http://127.0.0.1:8081/$MY_CART_UUID
```

Will receive headers back with the specific data needed. These are:

'X-Pacifica-Status' 'X-Pacifica-Message'

Message will be blank if there is no error. The list of possible status:

If the cart is waiting to be processed and there is no current state. "X-Pacifica-Status": "waiting"

If the cart is being processed and waiting for files to be staged locally. "X-Pacifica-Status": "staging"

If the cart has the files locally and is currently creating the tarfile. "X-Pacifica-Status": "bundling"

If the cart is finally ready for download. "X-Pacifica-Status": "ready"

If the cart has an error (such as no space available to create the tarfile). "X-Pacifica-Status": "error" "X-Pacifica-Message": "No Space Available"

3.1.3 Get a cart

To download the tarfile for the cart.

```
curl http://127.0.0.1:8081/$MY_CART_UUID?filename=my_cart.tar

In the above url my_cart.tar can be any file name of your choice
If no filename parameter is present you will get back data_date.tar in the form data_
↪YYYY_MM_DD_HH_MM_SS.tar
```

To save to file

```
curl -O -J http://127.0.0.1:8081/$MY_CART_UUID?filename=my_cart.tar

-O says to save to a file, and -J says to use the Content-Disposition file name the␣
↪server is trying to send back

Once this finishes there will be a tar file named my_cart.tar
Untar by:

tar xf my_cart.tar
```

3.1.4 Delete a Cart

Delete a created cart.

```
curl -X DELETE http://127.0.0.1:8081/$MY_CART_UUID
```

Data returned should be json telling you status of cart deletion.

3.2 Admin Commands

There is some interfaces to the internals of the carts via the admin command line interface `pacifica-cartd-cmd`.

3.2.1 Database Management

The command line interface has a couple of database management commands to verify the state of the database and whether it needs updating.

To check the current state of the database, run the following:

```
pacifica-cartd-cmd dbchk
echo $?
```

To update the database to the current version, run the following:

```
pacifica-cartd-cmd dbsync
echo $?
```

If either of these commands fail you may have issues connecting to the database configured. Be sure you are using the right configuration files for connecting to your database.

3.2.2 Rebuild or Fix a Cart

Sometimes carts will fail to build, build partially, or maybe you just want to handle cart building out of band. The command line interface has a way to do this.

```
pacifica-cartd-cmd fixit --cartid $MY_CART_UUID
```

3.2.3 Purge Old Carts that are older than a specific date.

Sometimes carts dont get cleaned up by users, and need to be expired. The command line interface has a way to do this.

```
pacifica-cartd-cmd purge --time-ago="60 days ago"
```

CHAPTER 4

Cartd Python Module

CHAPTER 5

Indices and tables

- genindex
- modindex
- search