Pacifica Cartd Documentation

David Brown

Feb 06, 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 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
```

3.2 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"

3.3 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
```

3.4 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.

CHAPTER 4

Cartd Python Module

4.1 Archive Requests Python Module

Module that is used by the cart to send requests to the archive interface.

class pacifica.cartd.archive_requests.**ArchiveRequests**
Class that supports all the requests to the archive interface.

__init__()
Constructor for setting the AI URL.

__weakref__
list of weak references to the object (if defined)

static _status_dict(*headers*, *file_name*)
Return status dictionary from http response headers.

pull_file(*archive_filename*, *cart_filepath*, *hashval*, *hashtype*)
Pull file from AI.

Performs a request that will attempt to write the contents of a file from the archive interface to the specified cart filepath

stage_file(*file_name*)
Send a post to the archive interface telling it to stage the file.

status_file(*file_name*)
Get a status from the archive interface via Head and returns response.

4.2 Configuration Python Module

Configuration reading and validation module.

pacifica.cartd.config.**get_config**()
Return the ConfigParser object with defaults set.

4.3 Globals Python Module

Used to load in all the carts environment variables.

Wrapped all in if statements so that they can be used in unit test environment

4.4 ORM Python Module

Cart Object Relational Model.

Using PeeWee to implement the ORM.

class pacifica.cartd.orm.**Cart**(**args*, ***kwargs*)
Cart object model.

DoesNotExist
alias of CartDoesNotExist

class pacifica.cartd.orm.**CartBase**(**args*, ***kwargs*)
Base Cart Model class.

DoesNotExist
alias of CartBaseDoesNotExist

classmethod atomic()
Do the DB atomic bits.

classmethod database_close()
Close the database connection.

classmethod database_connect()
Make sure database is connected.

Dont reopen connection.

reload()
Reload my current state from the DB.

class pacifica.cartd.orm.**CartSystem**(**args*, ***kwargs*)
Cart Schema Version Model.

DoesNotExist
alias of CartSystemDoesNotExist

classmethod get_or_create_version()
Set or create the current version of the schema.

classmethod get_version()
Get the current version as a tuple.

classmethod is_equal()
Check to see if schema version matches code version.

classmethod is_safe()
Check to see if the schema version is safe for the code.

class pacifica.cartd.orm.**File**(**args*, ***kwargs*)
File object model to keep track of what's been downloaded for a cart.

DoesNotExist
alias of FileDoesNotExist

class pacifica.cartd.orm.**OrmSync**
Special module for syncing the orm.

This module should incorporate a schema migration strategy.

The supported versions migrating forward must be in a versions array containing tuples for major and minor versions.

The version tuples are directly translated to method names in the orm_update class for the update between those versions.

Example Version Control:

```
class orm_update:
  versions = [
    (0, 1),
    (0, 2),
    (1, 0),
    (1, 1)
  ]

  def update_0_1_to_0_2():
      pass
  def update_0_2_to_1_0():
      pass
```

The body of the update should follow peewee migration practices. http://docs.peewee-orm.com/en/latest/peewee/playhouse.html#migrate

__weakref__
list of weak references to the object (if defined)

static create_tables()
Create the tables if they don't exist.

static dbconn_blocking()
Wait for the db connection.

classmethod update_0_1_to_1_0()
Update by adding the boolean column.

classmethod update_tables()
Update the database to the current version.

4.5 REST Python Module

Class for the cart interface.

Allows API to file interactions.

exception pacifica.cartd.rest.**CartInterfaceError**
CartInterfaceError.

Basic exception class for this module. Will be used to throw exceptions up to the top level of the application.

__weakref__
list of weak references to the object (if defined)

class pacifica.cartd.rest.**CartRoot**
Define the methods that can be used for cart request types.

Doctest for the cart generator class HPSS Doc Tests

static DELETE(*uid*)
Delete a cart that has been created.

static GET(*uid*, ***kwargs*)
Download the tar file created by the cart.

static HEAD(*uid*)
Get the status of a carts tar file.

static POST(*uid*)
Get all the files locally and bundled.

__weakref__
list of weak references to the object (if defined)

pacifica.cartd.rest.**bytes_type**(*unicode_obj*)
Convert the unicode object into bytes.

pacifica.cartd.rest.**error_page_default**(***kwargs*)
The default error page should always enforce json.

4.6 Celery Tasks Python Module

Module that contains all the amqp tasks that support the cart infrastructure.

4.7 Utilities Python Module

Module that has the utility functionality for the cart.

class pacifica.cartd.utils.**Cartutils**
Class used to provide utility functions for the cart to use.

__init__()
Default constructor setting environment variable defaults.

__weakref__
list of weak references to the object (if defined)

static available_cart(*uid*)
Check if the asked for cart tar is available.

Returns the path to tar if yes, false if not. None if no cart.

static cart_status(*uid*)
Get the status of a specified cart.

static check_file_modified_time(*response*, *cart_file*, *mycart*)
Check response for file modified time.

Should be from Archive Interface head request

check_file_ready_pull(*response*, *cart_file*, *mycart*)
Check file ready state.

Check response (should be from Archive Interface head request) for bytes per level then returns True or False based on if the file is at level 1 (downloadable)

static check_file_size_needed(*response*, *cart_file*, *mycart*)
Check response (should be from Archive Interface head request) for file size.

check_space_requirements(*cart_file*, *mycart*, *size_needed*, *deleted_flag*)
Check to make sure there is enough space available on disk for the file to be downloaded.

Note it will recursively call itself if there isnt enough space. It will delete a cart first, then call itself until either there is enough space or there is no carts to delete

check_status_details(*mycart*, *cart_file*, *size_needed*, *mod_time*)
Check to see if status response is correct.

Data from the status response is all correct and ready to for the file to be pulled.

static create_bundle_directories(*filepath*)
Create all the directories in the given path if they do not already exist.

classmethod create_download_path(*cart_file*, *mycart*, *abs_cart_file_path*)
Create the directories that the file will be pulled to.

delete_cart_bundle(*cart*)
Get the path to where a carts file are.

Also attempt to delete the file tree.

static fix_absolute_path(*filepath*)
Remove / from front of path.

classmethod get_path_size(*source*)
Return the size of a specific directory, including all subdirectories and files.

lru_cart_delete(*mycart*)
Delete the least recently used cart that isnt this one.

Only delete one cart per call.

prepare_bundle(*cartid*)
Check to see if all the files are staged locally.

Before calling the bundling action. If not will call itself to continue the waiting process

remove_cart(*uid*)
Call when a DELETE request comes in.

Verifies there is a cart to delete then removes it.

static set_file_status(*cart_file*, *cart*, *status*, *error*)
Set the status and/or error for a cart.

tar_files(*cartid*)
Start to bundle all the files together.

The option to do streaming download or not is based on a system configuration.

classmethod update_cart_files(*cart*, *file_ids*)
Update the files associated to a cart.

4.8 WSGI Python Module

Run the Cart Server. Cart module.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

C

D

E

F

G