Pacifica Python Downloader Documentation

David Brown

May 15, 2020

Contents:

The Pacifica Python Downloader library provides a Python API for downloading data from the Pacifica Core services. The Python Downloader allows users to build custom applications to download data from Pacifica.

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-downloader
```

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-downloader
```

CHAPTER 2

Code Examples

The Python Downloader has two architectural pieces. The first are a set of methods to manipulate metadata about files and generate a Cartd friendly metadata document. The second is a set of methods to interact with the Cartd service to wait and download the files.

2.1 Download Transaction Info

The download setup described below creates a temporary directory to download the data to.

```
from tempfile import mkdtemp

down_path = mkdtemp()
down = Downloader(cart_api_url='http://127.0.0.1:8081')
resp = requests.get('http://127.0.0.1:8181/status/transactions/by_id/67')
assert resp.status_code == 200
down.transactioninfo(down_path, resp.json())
```

2.2 Download Cloud Event

Often CloudEvents are handled in web server frameworks. Here's an example of using the downloader in CherryPy. This example can be launched as a consumer of CloudEvents sent by the Pacifica Notifications service.

```
from tempfile import mkdtemp
import cherrypy

class Root(object):
    exposed = True
    @cherrypy.tools.json_in()
    @cherrypy.tools.json_out()
    def POST(self):
```

(continues on next page)

(continued from previous page)

```
        """Accept the cloud event data and return the local download path."""
        down_path = mkdtemp()
        down = Downloader(cart_api_url='http://127.0.0.1:8081')
        down.cloudevent(down_path, cherrpy.request.json)
        return { 'download_path': down_path }


cherrypy.quickstart(Root(), '/', {
    '/': {
        'request.dispatch': cherrypy.dispatch.MethodDispatcher()
    }
})
```

CHAPTER 3

Downloader Python Module

3.1 Downloader Python Module

The Downloader internal Module.

class pacifica.downloader.downloader.**Downloader**(***kwargs*)

Downloader Class.

The other methods in this class are the supported download methods. Each method takes appropriate input for that method and the method will download the data to the location specified in the method's arguments.

__init__(***kwargs*)

Create the downloader.

Keyword arguments are delegated to the CartAPI.

_download_from_url(*location*, *cart_url*, *filename*)

Download the cart from the url.

The cart url is returned from the CartAPI.

cloudevent(*location*, *cloudevent*, ***kwargs*)

Handle a cloud event and download the data in a cart.

CloudEvents is a specification for passing information about changes in cloud infrastructure or state. This method consumes events produced by the Pacifica Notifications service.

transactioninfo(*location*, *transinfo*, ***kwargs*)

Handle transaction info and download the data in a cart.

Transaction info objects are pulled from the PolicyAPI.

3.2 CartAPI Python Module

Cart API module for interacting with carts.

class pacifica.downloader.cartapi.**CartAPI**(***kwargs*)
Cart api object for manipulating carts.

This class has two methods used for setting up a cart and waiting for completion.

__init__(***kwargs*)
Constructor for cart api.

The constructor takes a required URL to the Cart API. Optionally, there can be passed a requests session via keyword arguments. Also, an optional requests authentication dictionary can be passed via keyword arguments.

_addr = None

_auth = None

_cart_api_url = None

_extra_args = None

_port = None

_proto = None

auth
Return the requests authentication dictionary.

cart_api_url
Return the CartAPI URL.

setup_cart(*yield_files*)
Setup a cart from the method and return url to the download.

This method takes a callable argument that returns an iterator. The iterator is used to generate a list that is directly sent to the Cartd API. This method returns the full url to the cart created.

wait_for_cart(*cart_url*, *timeout=120*)
Wait for cart completion to finish.

This method takes a cart url returned from the *setup_cart()* method and polls the endpoint until the cart is ready to download.

3.3 CloudEvent Python Module

Cloud Event Parser.

class pacifica.downloader.cloudevent.**CloudEvent**
Cloud Event Parser.

static yield_files(*cloudevent*)
Returned a method for yield files.

The cloud event passed contains a 'data' key that is a flat list of metadata objects. Some of those objects are destined for the 'Files' table.

3.4 Policy Python Module

Policy Parser.

class pacifica.downloader.policy.**TransactionInfo**
Cloud Event Parser.

static yield_files(*transinfo*)
Return a method for yield files.

The files are part of a 'files' key that contains a dictionary that is keyed off the files ID.

Pacifica Downloader Module.

The primary exposed class is the *Downloader* class. There are two internal classes to pull the metadata required to interact with the Cartd service.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

C

D

P

S

T

W

Y