Pacifica Ingest Documentation

David Brown

Jan 26, 2020

Contents:

The Pacifica Ingest service provides endpoints to consume data and metadata for ingest into 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-ingest
```

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-ingest
```

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 Ingest 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: 8066

[/]
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:

```
[ingest]
; This section is specific to the ingest processes

; Local directory for incoming data and metadata
volume_path = /tmp
```

(continues on next page)

(continued from previous page)

```
[uniqueid]
; This section describes where the UniqueID service is

; URL to the endpoint
url = http://127.0.0.1:8051

[policy]
; This section describes what endpoints are on the policy service

; Ingest URL to verify metadata
ingest_url = http://127.0.0.1:8181/ingest

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

; URL to the endpoint
url = http://127.0.0.1:8080

[metadata]
; This section describes what endpoints are on the metadata service

; Ingest URL for ingest metadata
ingest_url = http://127.0.0.1:8121/ingest

[celery]
; This section contains celery messaging configuration

; The broker url is how messages get passed around
broker_url = pyamqp://

; The backend url is how return results are sent around
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 Ingest 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 Ingest service is more critical for uploaders than the rest of the Pacifica Core services. This is the first service that must be put on the edge of your infrastructure and closest to where you are getting your data from.

2.3.2 CherryPy Server

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

```
$ pacifica-ingest --help
usage: pacifica-ingest [-h] [--cp-config CPCONFIG] [-c CONFIG] [-p PORT]
                       [-a ADDRESS]

Run the cart server.

optional arguments:
  -h, --help            show this help message and exit
  --cp-config CPCONFIG  cherrypy config file
  -c CONFIG, --config CONFIG
                        ingest config file
  -p PORT, --port PORT  port to listen on
  -a ADDRESS, --address ADDRESS
                        address to listen on
$ pacifica-ingest-cmd dbsync
$ pacifica-ingest
[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://0.0.0.0:8066
[09/Jan/2019:09:17:26] ENGINE Bus STARTED
```

2.3.3 uWSGI Server

To make running the Ingest 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 :8066 --master --module pacifica.ingest.wsgi
```

CHAPTER 3

Example Usage

The first thing to discuss when talking about interacting with the ingest service is data format.

3.1 Bundle Format

The bundle format is parsed using the tarfile package from the Python standard library.

Both data and metadata are stored in a bundle. Metadata is stored in the `metadata.txt` file (JSON format). Data is stored in the `data/` directory.

To display the contents of a bundle using the `tar` command:

```
tar -tf mybundle.tar
```

For example, the contents of `mybundle.tar` is:

```
data/mywork/project/project.doc
data/mywork/experiment/results.csv
data/mywork/experiment/results.doc
metadata.txt
```

3.2 API Examples

The endpoints that define the ingest process are as follows. The assumption is that the installer knows the IP address and port the WSGI service is listening on.

3.2.1 Ingest (Single HTTP Request)

Post a bundle (*defined above*) to the endpoint.

```
POST /ingest
... tar bundle as body ...
```

The response will be the job ID information as if you requested it directly.

```
{
  "job_id": 1234,
  "state": "OK",
  "task": "UPLOADING",
  "task_percent": "0.0",
  "updated": "2018-01-25 16:54:50",
  "created": "2018-01-25 16:54:50",
  "exception": ""
}
```

Failures that exist with this endpoint are during the course of uploading the bundle. Sending data to this endpoint should consider long drawn out HTTP posts that maybe longer than clients are used to handling.

3.2.2 Move (Single HTTP Request)

Post a metadata document to the endpoint.

```
POST /move
... content of move-md.json ...
```

The response will be the job ID information as if you requested it directly.

```
{
  "job_id": 1234,
  "state": "OK",
  "task": "UPLOADING",
  "task_percent": "0.0",
  "updated": "2018-01-25 16:54:50",
  "created": "2018-01-25 16:54:50",
  "exception": ""
}
```

3.2.3 Get State for Job

Using the `job_id` field from the HTTP response from an ingest.

```
GET /get_state?job_id=1234
{
  "job_id": 1234,
  "state": "OK",
  "task": "ingest files",
  "task_percent": "0.0",
  "updated": "2018-01-25 17:00:32",
  "created": "2018-01-25 16:54:50",
  "exception": ""
}
```

As the bundle of data is being processed errors may occure, if that happens the following will be returned. It is useful when consuming this endpoint to plan for failures. Consider logging or showing a message visable to the user that shows the ingest failed.

```
GET /get_state?job_id=1234
{
  "job_id": 1234,
  "state": "FAILED",
  "task": "ingest files",
  "task_percent": "0.0",
  "updated": "2018-01-25 17:01:02",
  "created": "2018-01-25 16:54:50",
  "exception": "... some crazy python back trace ..."
}
```

CHAPTER 4

CLI Tools

There is an admin tool that consists of subcommands for manipulating ingest processes.

4.1 Job Subcommand

The job subcommand allows administrators to directly manipulate the state of a job. Due to complex computing environments some jobs may get "stuck" and get to a state where they aren't failed and aren't progressing. This may happen for any number of reasons but the solution is to manually fail the job.

```
IngestCMD job \
    --job-id 1234 \
    --state FAILED \
    --task 'ingest files' \
    --task-percent 0.0 \
    --exception 'Failed by adminstrator'
```

CHAPTER 5

Ingest Python Module

5.1 Configuration Python Module

Configuration reading and validation module.

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

5.2 Globals Python Module

Global configuration options expressed in environment variables.

5.3 ORM Python Module

ORM for index server.

class `pacifica.ingest.orm.`**`BaseModel`**`(`**args*, ***kwargs*`)`
Auto-generated by pwiz.

`DoesNotExist`
alias of `BaseModelDoesNotExist`

`_meta = <peewee.Metadata object>`

`_schema = <peewee.SchemaManager object>`

`id = <AutoField: BaseModel.id>`

class `pacifica.ingest.orm.`**`IngestState`**`(`**args*, ***kwargs*`)`
Map a python record to a mysql table.

`DoesNotExist`
alias of `IngestStateDoesNotExist`

_meta = <peewee.Metadata object>

_schema = <peewee.SchemaManager object>

classmethod atomic()
Get the atomic context or decorator.

complete = <BooleanField: IngestState.complete>

created = <DateTimeField: IngestState.created>

classmethod database_close()
Close the database connection.

Closing already closed database is not a problem, so continue on.

classmethod database_connect()
Make sure database is connected.

Trying to connect a second time *does* cause problems.

exception = <TextField: IngestState.exception>

job_id = <BigIntegerField: IngestState.job_id>

state = <CharField: IngestState.state>

task = <CharField: IngestState.task>

task_percent = <DecimalField: IngestState.task_percent>

updated = <DateTimeField: IngestState.updated>

class pacifica.ingest.orm.**IngestStateSystem**(**args*, ***kwargs*)
Ingest State Schema Version Model.

DoesNotExist
alias of `IngestStateSystemDoesNotExist`

_meta = <peewee.Metadata object>

_schema = <peewee.SchemaManager object>

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.

part = <CharField: IngestStateSystem.part>

value = <IntegerField: IngestStateSystem.value>

class pacifica.ingest.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

static dbconn_blocking()
Wait for the db connection.

classmethod update_0_0_to_1_0()
Update by creating the table.

classmethod update_1_0_to_2_0()
Update by adding the boolean column.

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

versions = [(0, 0), (1, 0), (2, 0)]

pacifica.ingest.orm.**read_state**(*job_id*)
Return the state of an ingest job as a json object.

pacifica.ingest.orm.**update_state**(*job_id*, *state*, *task*, *task_percent*, *exception=''*)
Update the state of an ingest job.

5.4 REST Python Module

Ingest Server Main.

class pacifica.ingest.rest.**RestIngestState**
The CherryPy ingest state object.

static GET(*job_id*)
Get the ingest state for the job.

exposed = True

class pacifica.ingest.rest.**RestMove**
Ingest the data from the service.

static POST()
Post the uploaded data.

exposed = True

class pacifica.ingest.rest.**RestUpload**
Ingest the data from the service.

static POST()
Post the uploaded data.

exposed = True

class pacifica.ingest.rest.**Root**
The CherryPy root object.

exposed = False

get_state = <pacifica.ingest.rest.RestIngestState object>

move = <pacifica.ingest.rest.RestMove object>

upload = <pacifica.ingest.rest.RestUpload object>

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

5.5 Tar Utilities Python Module

Utilities and classes for unbundling and archiving a tar file.

class pacifica.ingest.tarutils.**FileIngester**(*hashtype*, *hashcode*, *file_id*)
Class to ingest a single file from a tar file into the file archives.

__init__(*hashtype*, *hashcode*, *file_id*)
Constructor for FileIngester class.

file_id = 0

fileobj = None

hashval = None

read(*size*)
Read wrapper for requests that calculates the hashcode inline.

recorded_hash = ''

server = ''

upload_file_in_file(*info*, *tar*)
Upload a file from inside a tar file.

validate_hash()
Validate that the calculated hash matches the hash uploaded in the tar file.

exception pacifica.ingest.tarutils.**HashValidationException**
Class to capture hashsum validation failures.

class pacifica.ingest.tarutils.**MetaParser**
Class used to hold and search metadata.

__init__()
Constructor.

clean_metadata()
clean /data from filepaths.

file_count = -999

file_obj_count (*meta_list*)
Count the file objects in metadata and keep the count.

files = {}

get_fname (*file_id*)
Get the file name from the file ID.

get_hash (*file_id*)
Return the hash string for a file name.

get_subdir (*file_id*)
Get the sub directory element from the file ID.

load_meta (*tar*, *job_id*)
Load the metadata from a tar file into searchable structures.

meta = None

meta_str = ''

post_metadata ()
Upload metadata to server.

read_meta (*metafile*, *job_id*)
Read the metadata from metafile and assume it's good.

start_id = -999

transaction_id = -999

class pacifica.ingest.tarutils.**TarIngester** (*tar*, *meta*)
Class to read a tar file and upload it to the metadata and file archives.

__init__ (*tar*, *meta*)
Constructor for TarIngester class.

ingest ()
Ingest a tar file into the file archive.

meta = None

tar = None

pacifica.ingest.tarutils.**file_count** (*tar*)
Retrieve the file count for a tar file.

Does not count metadata.txt as that is not uploaded to the file archive

pacifica.ingest.tarutils.**get_clipped** (*fname*)
Return a file path with the data separator removed.

pacifica.ingest.tarutils.**open_tar** (*fpath*)
Seek to the location of fpath, returns a file stream pointer and file size.

pacifica.ingest.tarutils.**patch_files** (*meta_obj*)
Patch the files in the archive interface.

5.6 Celery Tasks Python Module

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

exception pacifica.ingest.tasks.**IngestException**
Ingest class exception.

pacifica.ingest.tasks.**ingest_check_tarfile**(*job_id*, *filepath*)
Check the ingest tarfile and return state or set it properly.

pacifica.ingest.tasks.**ingest_files**(*job_id*, *ingest_obj*)
Ingest the files to the archive interface.

pacifica.ingest.tasks.**ingest_metadata**(*job_id*, *meta*)
Ingest metadata to the metadata service.

pacifica.ingest.tasks.**ingest_metadata_parser**(*job_id*, *tar*)
Ingest the metadata and set the state appropriately.

pacifica.ingest.tasks.**ingest_policy_check**(*job_id*, *meta_str*)
Ingest check to validate metadata at policy.

pacifica.ingest.tasks.**move_files**(*job_id*, *meta_obj*)
Move the files to the archive interface.

pacifica.ingest.tasks.**move_metadata_parser**(*job_id*, *metafile*)
Ingest the metadata and set the state appropriately.

pacifica.ingest.tasks.**validate_meta**(*meta_str*)
Validate metadata.

5.7 Utilities Python Module

Testable utilities for ingest.

pacifica.ingest.utils.**create_state_response**(*record*)
Create the state response body from a record.

pacifica.ingest.utils.**get_unique_id**(*id_range*, *mode*)
Return a unique job id from the id server.

pacifica.ingest.utils.**parse_size**(*size*)
Parse size string to integer.

5.8 WSGI Python Module

The WSGI interface module for notifications. Ingest module.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V