Pacifica Notifications Documentation

David Brown

Feb 08, 2019

Contents:

This service is a Pacifica Policy based routing mechanism for data subscribers to execute workflows based on the availability of data in 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-notifications
```

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-notifications
```

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 Notifications 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: 8070

[/]
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:

```
[notifications]
; This section describes notification specific configurations

; The policy server endpoint to query
policy_url = http://127.0.0.1:8181
```

(continues on next page)

(continued from previous page)

```
[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 Notifications 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 Notifications service is relatively new and has not seen usage enough to know how it performs.

2.3.2 CherryPy Server

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

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

Run the notifications server.

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        cherrypy config file
```

(continues on next page)

(continued from previous page)

```
  -p PORT, --port PORT  port to listen on
  -a ADDRESS, --address ADDRESS
                        address to listen on
$ pacifica-notifications-cmd dbsync
$ pacifica-notifications
[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://0.0.0.0:8070
[09/Jan/2019:09:17:26] ENGINE Bus STARTED
```

2.3.3 uWSGI Server

To make running the Notifications 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 :8070 --master --module pacifica.notifications.wsgi
```

CHAPTER 3

Example Usage

The (Pacifica Metadata)[https://github.com/pacifica/pacifica-metadata.git] service emits (CloudEvents)[https://github.com/cloudevents/spec] when new data is accepted. This service is intended to recieve and route those events to users that are allowed based on Pacifica Policy.

3.1 API Reference

There are two REST APIs available on this service. The first api accepts cloud events for processing. The second api allows users to subscribe to events and specify routing target urls to send those events.

3.1.1 Cloud Events Recieve

```
POST /receive
Content-Type: application/json
... JSON Cloud Event ...
```

3.1.2 Subscriptions

The subscriptions API is a REST style API accessed on `/eventmatch`.

Create Event Subscription

Request:

```
POST /eventmatch
Http-Remote-User: dmlb2001
Content-Type: application/json
{
```

(continues on next page)

(continued from previous page)

```
  "name": "My Event Match",
  "jsonpath": "data",
  "target_url": "http://www.example.com/recieve"
}
```

Response:

```
Content-Type: application/json
{
  "user": "dmlb2001",
  "updated": "2018-08-02T13:53:05.838827",
  "uuid": "466725b0-cbe1-45cd-b034-c3209aa4b6e0",
  "deleted": null,
  "version": "v0.1",
  "jsonpath": "data",
  "disabled": null,
  "created": "2018-08-02T13:53:05.838827",
  "name": "My Event Match",
  "extensions": {},
  "target_url": "http://www.example.com/receive",
  "error": null
}
```

Get Event Subscription

Request:

```
GET /eventmatch/466725b0-cbe1-45cd-b034-c3209aa4b6e0
Http-Remote-User: dmlb2001
Content-Type: application/json
```

Response:

```
Content-Type: application/json
{
  "user": "dmlb2001",
  "updated": "2018-08-02T13:53:05.838827",
  "uuid": "466725b0-cbe1-45cd-b034-c3209aa4b6e0",
  "deleted": null,
  "version": "v0.1",
  "jsonpath": "data",
  "disabled": null,
  "created": "2018-08-02T13:53:05.838827",
  "name": "My Event Match",
  "extensions": {},
  "target_url": "http://www.example.com/receive",
  "error": null
}
```

Update Event Subscription

Request:

```
PUT /eventmatch/466725b0-cbe1-45cd-b034-c3209aa4b6e0
Http-Remote-User: dmlb2001
Content-Type: application/json
{
  "target_url": "http://api.example.com/receive"
}
```

Response:

```
Content-Type: application/json
{
  "user": "dmlb2001",
  "updated": "2018-08-02T13:53:05.838827",
  "uuid": "466725b0-cbe1-45cd-b034-c3209aa4b6e0",
  "deleted": null,
  "version": "v0.1",
  "jsonpath": "data",
  "disabled": null,
  "created": "2018-08-02T13:53:05.838827",
  "name": "My Event Match",
  "extensions": {},
  "target_url": "http://api.example.com/receive",
  "error": null
}
```

Delete Event Subscription

Request:

```
DELETE /eventmatch/466725b0-cbe1-45cd-b034-c3209aa4b6e0
```

Response:

```
HTTP/1.1 200 OK
```

CHAPTER 4

Notifications Python Module

4.1 Configuration Python Module

Configuration reading and validation module.

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

4.2 Globals Python Module

Global configuration options expressed in environment variables.

4.3 ORM Python Module

The ORM module defining the SQL model for notifications.

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

DoesNotExist
alias of BaseModelDoesNotExist

classmethod atomic()
Do the database atomic action.

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.

class pacifica.notifications.orm.**EventMatch**(**args*, ***kwargs*)
Events matching via jsonpath per user.

DoesNotExist
alias of EventMatchDoesNotExist

to_hash()
Convert the object to a json serializable hash.

validate_jsonpath()
Validate the jsonpath string.

class pacifica.notifications.orm.**NotificationSystem**(**args*, ***kwargs*)
Notification Schema Version Model.

DoesNotExist
alias of NotificationSystemDoesNotExist

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.notifications.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 dbconn_blocking()
Wait for the db connection.

classmethod update_0_0_to_1_0()
Update by adding the boolean column.

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

4.4 REST Python Module

CherryPy module containing classes for rest interface.

class pacifica.notifications.rest.**EventMatch**
CherryPy EventMatch endpoint.

classmethod DELETE(*event_uuid*)
Delete the event by uuid.

classmethod GET(*event_uuid=None*)
Get the event ID and return it.

classmethod POST()
Create an Event Match obj in the database.

classmethod PUT(*event_uuid*)
Update an Event Match obj in the database.

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

static _http_get(*event_uuid*)
Internal get event by UUID and return peewee obj.

class pacifica.notifications.rest.**ReceiveEvent**
CherryPy Receive Event object.

classmethod POST()
Receive the event and dispatch it to backend.

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

class pacifica.notifications.rest.**Root**
CherryPy Root Object.

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

pacifica.notifications.rest.**encode_text**(*thing_obj*)
Encode the text to bytes.

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

pacifica.notifications.rest.**get_remote_user**()
Get the remote user from cherrypy request headers.

4.5 JSON Path Python Module

The jsonpath interface module.

pacifica.notifications.jsonpath.**find**(*expr*, *data*)
Match the expression in the data.

pacifica.notifications.jsonpath.**parse**(*jsonpath_str*)
Parse the json path.

4.6 Celery Tasks Python Module

The Celery tasks module.

pacifica.notifications.tasks.**disable_eventmatch**(*eventmatch_uuid*, *error*)
Disable the eventmatch obj.

4.7 WSGI Python Module

The WSGI interface module for notifications. Pacifica Notifications Module.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

D

E

F

G

I

N

O

P

R

T

U

V