rapidsms-celery-router Documentation

Release 0.0.1

RapidSMS development community

November 06, 2013

Contents

rapidsms-celery-router is a custom RapidSMS router implementation that uses Celery to queue incoming and outgoing messages.

Warning: rapidsms-celery-router is only compatible with the feature/new-routing branch of RapidSMS.

CHAPTER 1

Motivation

RapidSMS ships with a router, `BlockingRouter`, that processes messages synchronously in the main HTTP thread. This is fine for most scenarios, but in some cases you may wish to process messages outside of the HTTP request/response cycle to be more efficient. rapidsms-celery-router is a custom router that allows you queue messages for background processing. It's designed for projects that require high messages volumes and greater concurrency.

CHAPTER 2

Contents

2.1 Installation

2.1.1 Dependencies

rapidsms-celery-router depends on django-celery. Please follow the django-celery setup instructions before proceeding with these steps.

2.1.2 Setup

Install rapidsms-celery-router using Pip:

```
pip install rapidsms-celery-router
```

This will install the necessary dependencies, including django-celery and celery, if required.

Set `RAPIDSMS_ROUTER` in your project's `settings.py` to use `CeleryRouter`:

```
RAPIDSMS_ROUTER = "celery_router.router.CeleryRouter"
```

That's it! Now all incoming and outgoing messages will be queued using Celery.

2.2 Configuration

2.2.1 Eager backends

Sometimes your project may require the use of a synchronous backend. If this is the case, you can configure specific backends to utilize Celery's eager functionality with the `celery_router.eager` backend setting. For example, here's how you can force the httptester backend to be eager:

```
INSTALLED_BACKENDS = {
    "message_tester": {
        "ENGINE": "rapidsms.contrib.httptester.backend",
        "celery_router.eager": True, # <----------
    },
}
```

Using this setting means that the task will be executed in the current process, and not by an asynchronous worker. Please see the Celery documentation for more information on calling tasks.

2.2.2 Logging

Note: Please see the Django logging documentation for further information regarding general logging configuration.

All logging specific to rapidsms-celery-router is handled through the `celery_router` name. For example, if you have a `file` handler defined, you can capture all messages using the following configuration:

```
LOGGING_CONFIG = {
    'celery_router': {
        'handlers': ['file'],
        'level': 'DEBUG',
    },
}
```

Currently, there are only two child loggers: one for the router and one for the Celery task. You can capture their messages independently like so:

```
LOGGING_CONFIG = {
    'celery_router.router': {
        'handlers': ['file'],
        'level': 'INFO',
    },
    'celery_router.tasks.rapidsms_handle_message': {
        'handlers': ['file'],
        'level': 'DEBUG',
    },
}
```

BlockingRouter

rapidsms-celery-router's tasks use the `BlockingRouter` to route messages. If you want to capture all router messages, make sure to add, in addition to the `celery_router` loggers, `blockingrouter`:

```
LOGGING_CONFIG = {
    'blockingrouter': {
        'handlers': ['file'],
        'level': 'DEBUG',
    }
}
```

2.3 Running the tests

You can run the tests with via:

```
python setup.py test
```

or:

```
python runtests.py
```

2.4 Release Notes

2.4.1 v0.0.1 (09/24/2012)

- Initial release