Seed Stage-Based Messaging Store Documentation

Release 0.10.1

Praekelt.org

Oct 18, 2018

Contents

1 Seed Stage-Based Messaging Store documentation **1**
- 1.1 Getting started . . . 1

2 Requirements **3**
- 2.1 Overview . . . 3
- 2.2 Python requirements . . . 3
- 2.3 Seed Requirements . . . 3

3 Setup **5**
- 3.1 Installing . . . 5
- 3.2 Configuration Options . . . 5

4 Data Models **7**
- 4.1 Content . . . 7
- 4.2 Subscriptions . . . 8

5 Authentication and Authorization **11**
- 5.1 Basics . . . 11
- 5.2 Users and Groups . . . 11
- 5.3 Authorization and permissions . . . 11

6 API Details **13**
- 6.1 Authenticating to the API . . . 13
- 6.2 Pagination . . . 13
- 6.3 Endpoints . . . 14

7 Integrations with other Seed services **19**
- 7.1 Seed Scheduler . . . 19
- 7.2 Seed Message Sender . . . 19
- 7.3 Project Hub . . . 20

8 Production requirements and setup **21**
- 8.1 Running in Production . . . 21

9 Indices, glossary and tables **23**

HTTP Routing Table **25**

CHAPTER 1

Seed Stage-Based Messaging Store documentation

The Seed Stage-Based Messaging Store is one of the microservices in the Seed Stack.

The Stage-Based Messaging Store has the following key responsibilities:

- Store the stage-based content (both audio and text).
- Store the stage-based content schedules.
- Store the stage-based content subscriptions for each user.

Getting started

The following resources are provided to help you get started running or developing the Seed Stage-Based Messaging Store:

- Learn about the *Requirements* for running the service and basic *Setup* instructions.
- Read about the *Data Models* used by the service.
- Read about the *Authorization* requirements.
- Browse the *API Documentation* for the available endpoints and parameters.
- Read about the *Integrations* this service has with the other Seed Stack services.
- Learn about what is required when running the service in *Production*

CHAPTER 2

Requirements

Overview

The Seed Stage-Based Messaging Store requires the following dependencies to run:

- Python 2.7
- PostgreSQL >= 9.3
- Redis >= 2.10 or RabbitMQ >= 3.4 as the Celery Broker

Python requirements

The full list of Python packages required are detailed in the project's setup.py file, but the major ones are:

- Django 1.9
- Django REST Framework 3.3
- Celery 3.1

Note: A celery worker needs to be running to process post-save tasks and scheduled metric firing tasks.

Seed Requirements

The Seed Stage-Based Messaging Store only depends on one other seed service, the Go Metrics API.

CHAPTER 3

Setup

Installing

The steps required to install the Seed Stage-based Messaging Service are:

1. Get the code from the Github Project with git:

   ```
   $ git clone https://github.com/praekelt/seed-stage-based-messaging.git
   ```

   This will create a directory `seed-stage-based-messaging` in your current directory.

1. Install the Python requirements with pip:

   ```
   $ pip install -r requirements.txt
   ```

   This will download and install all the Python packages required to run the project.

2. Setup the database:

   ```
   $ python manage migrate
   ```

   This will create all the database tables required.

   Note: The PostgreSQL database for the Seed Stage-based Messaging Store needs to exist before running this command. See *STAGE_BASED_MESSAGING_DATABASE* for details.

3. Run the development server:

   ```
   $ python manage.py runserver
   ```

   Note: This will run a development HTTP server. This is only suitable for testing and development, for production usage please see *Running in Production*

Configuration Options

The main configuration file is `seed_stage_based_messaging/settings.py`.

The following environmental variables can be used to override some default settings:

SECRET_KEY
This overrides the Django `SECRET_KEY` setting.

DEBUG
This overrides the Django `DEBUG` setting.

USE_SSL
Whether to use SSL when build absolute URLs. Defaults to False.

STAGE_BASED_MESSAGING_DATABASE
The database parameters to use as a URL in the format specified by the DJ-Database-URL format.

STAGE_BASED_MESSAGING_SENTRY_DSN
The DSN to the Sentry instance you would like to log errors to.

BROKER_URL
The Broker URL to use with Celery.

STAGE_BASED_MESSAGING_URL
The URL of the instance of the Seed Stage-based Messaging API that will be used when creating POST-back hooks to this service from other Seed services.

SCHEDULER_URL
The URL to the Seed Scheduler API instance.

SCHEDULER_API_TOKEN
The *auth token* to use to connect to the Seed Scheduler API instance above.

SCHEDULER_INBOUND_API_TOKEN
The *auth token* to use to connect to this Seed Stage-based Messaging API from POST-backs from the Seed Scheduler API instance.

IDENTITY_STORE_URL
The URL to the Seed Identity Store API instance.

IDENTITY_STORE_TOKEN
The *auth token* to use to connect to the Seed Identity Store API instance above.

MESSAGE_SENDER_URL
The URL to the Seed Message Sender API instance.

MESSAGE_SENDER_TOKEN
The *auth token* to use to connect to the Seed Message Sender API instance above.

METRICS_URL
The URL to the Go Metrics API instance to push metrics to.

METRICS_AUTH_TOKEN
The *auth token* to use to connect to the Go Metrics API above.

CHAPTER 4

Data Models

Content

Schedule

Represents either a fixed date & time or interval based cron-like schedule.

Fields

id An auto incrementing integer unique identifier for the record.

minute A character field representing the minute portion of the schedule. Defaults to * but can be a comma separated list of numbers between 0 and 59.

hour A character field representing the hour portion of the schedule. Defaults to * but can be a comma separated list of numbers between 0 and 23.

day_of_week A character field representing the day of the week portion of the schedule. Defaults to * but can be a comma separated list of numbers between 1 and 7 where 1 = Monday and 7 = Sunday.

day_of_month A character field representing the day of the month portion of the schedule. Defaults to * but can be a comma separated list of numbers between 1 and 31.

month_of_year A character field representing the month of the year portion of the schedule. Defaults to * but can be a comma separated list of numbers between 1 and 12.

MessageSet

Represents a group of Messages a recipient can be sent and the default Schedule they can be sent on.

Fields

id An auto incrementing integer unique identifier for the record.

short_name A unique name that identifies this MessageSet.

notes An optional free text field for notes about this MessageSet.

next_set An optional self-referencing link to a MessageSet that should follow from this one.

default_schedule A reference to a Schedule used as the default for this MessageSet.

content_type A choice field between *audio* and *text* representing the type of content this MessageSet contains.

created_at A date and time field of when the record was created.

updated_at A date and time field of when the record was last updated.

BinaryContent

Represents binary file storage for use in the Message object.

Fields

id An auto incrementing integer unique identifier for the record.

content A FileField that represents the binary content's location on disk.

created_at A date and time field of when the record was created.

updated_at A date and time field of when the record was last updated.

Message

Represents a Message that a recipient can be sent. Can be either text-based or binary (audio) and is language specific.

Fields

id An auto incrementing integer unique identifier for the record.

messageset A reference to the MessageSet this message belongs to.

sequence_number A required integer representing the order of this message in the set.

lang An ISO639-3 language code.

text_content Optional text content for the message.

binary_content Optional reference to a BinaryContent.

created_at A date and time field of when the record was created.

updated_at A date and time field of when the record was last updated.

Subscriptions

Subscription

Represents a specific Identities subscription to a MessageSet, including the shedule and current state.

Fields

id A UUID 4 unique identifier for the record.

identity A UUID reference to an Identity stored in the Seed Identity Store.

version An integer version number of the Subscription schema used.

messageset A reference to the MessageSet for this Subscription.

next_sequence_number The integer Message sequence number to use for this Subscription.

lang An ISO639-3 language code representing the preferred language for this Subscription.

active A boolean of the active status.

completed A boolean of the complete status.

schedule A reference to the Schedule to use for this Subscription.

process_status A integer flag representing the process status of this subscription.

-2 = error
-1 = error
0 = ready
1 = in process
2 = completed

metadata A JSON field of *metadata* to be stored with the Subscription.

created_at A date and time field of when the record was created.

updated_at A date and time field of when the record was last updated.

created_by A reference to the User account that created this record.

updated_by A reference to the User account that last updated this record.

CHAPTER 5

Authentication and Authorization

Basics

Authentication to the Seed Stage-Based Messaging Store API is provided the Token Authentication feature of the Django REST Framework.

In short, each user of this API needs have been supplied a unique secret token that must be provided in the `Authorization` HTTP header of every request made to this API.

An example request with the `Authorization` header might look like this:

```
POST /endpoint/ HTTP/1.1
Host: <stage-based-messaging-store-domain>
Content-Type: application/json
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Users and Groups

User and *Group* objects are provided by the Django Auth framework and can be added and created through the normal maintenance methods (Django Admin, Dgango Shell, ...).

There is also a rudimentary API endpoint: *`POST /user/token/`* that will create a user and token for a given email address (or just a token if a user with that email address already exists).

Authorization and permissions

All of the current API endpoints do not require any specific permissions other than a valid authenticated user.

The only exception to this is *`POST /user/token/`* which requires an admin level user.

CHAPTER 6

API Details

The Seed Stage-Based Messaging Store provides REST like API with JSON payloads.

The root URL for all of the endpoints is:

```
https://<stage-based-messaging-store-domain>/api/
```

Authenticating to the API

Please see the *Authentication and Authorization* document.

Pagination

When the results set is larger than a configured amount, the data is broken up into pages using the limit and offset parameters.

Paginated endpoints will provide information about the total amount of items available along with links to the previous and next pages (where available) in the returned JSON data.

GET / (*any*) **/**

Query Parameters

- **limit** – the amount of record to limit a page of results to.
- **offset** – the starting position of the query in relation to the complete set of unpaginated items

Response JSON Object

- **count** (*int*) – the total number of results available
- **previous** (*string*) – the URL to the previous page of results (if available)
- **next** (*string*) – the URL to the next page of results (if available)

Example request:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "count": 50,
```

```
    "next": "http://sbm.example.org/api/v1/enpoint/?limit=10&offset=30",
    "previous": "http://smb.example.org/api/v1/endpoint/?limit=10&offset=10",
    "results": []
}
```

Endpoints

The endpoints provided by the Seed Stage-Based Messaging Store are split into two categories, core endpoints and helper endpoints

Core

The root URL for all of the core endpoints includes the version prefix (`https://<stage-based-messaging-store-domain>/api/v1/`)

`POST /user/token/`

Creates a user and token for the given email address.

If a user already exists for the given email address, the existing user account is used to generate a new token.

Request JSON Object

- **`email`** (*`string`*) – the email address of the user to create or use.

Response JSON Object

- **`token`** (*`string`*) – the auth token generated for the given user.

Status Codes

- 201 Created – token successfully created.
- 400 Bad Request – an email address was not provided or was invalid.
- 401 Unauthorized – the token is invalid/missing.

Example request:

```
POST /user/token/ HTTP/1.1
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b

{
    "email": "bob@example.org"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "token": "c05fbab6d5f912429052830c77eeb022249324cb"
}
```

Content

GET /schedule/
Returns a list of Schedules.

POST /schedule/
Creates a new Schedule.

GET /schedule/ (**int:** *schedule_id*) **/**
Retuns the Schedule record for a given schedule_id.

PUT /schedule/ (**int:** *schedule_id*) **/**
Updates the Schedule record for a given schedule_id.

DELETE /schedule/ (**int:** *schedule_id*) **/**
Deletes the Schedule record for a given schedule_id.

GET /messageset/
Returns a list of MessageSets.

POST /messageset/
Creates a new MessageSet.

GET /messageset/ (**int:** *messageset_id*) **/**
Retuns the MessageSet record for a given messageset_id.

PUT /messageset/ (**int:** *messageset_id*) **/**
Updates the MessageSet record for a given messageset_id.

DELETE /messageset/ (**int:** *messageset_id*) **/**
Deletes the MessageSet record for a given messageset_id.

GET /messageset/ (**int:** *messageset_id*) **/messages/**
Returns a list of Messages for a given messageset_id.

GET /message/
Returns a list of Messages.

POST /message/
Create a new Message record.

GET /message/ (**int:** *message_id*) **/**
Returns the Message record for a given message_id.

PUT /message/ (**int:** *message_id*) **/**
Updates the Message record for a given message_id.

DELETE /message/ (**int:** *message_id*) **/**
Deletes the Message record for a given message_id.

GET /message/ (**int:** *message_id*) **/content/**
Returns the content for a given message_id.

GET /binarycontent/
Returns a list of BinaryContent records.

POST /binarycontent/
Creates a new BinaryContent record.

GET /binarycontent/ (**int:** *binarycontent_id*) **/**
Returns the BinaryContent record for a given binarycontent_id.

PUT /binarycontent/ (**int:** *binarycontent_id*) **/**
Updates the BinaryContent record for a given binarycontent_id.

DELETE /binarycontent/ (**int:** *binarycontent_id*) **/**
Deletes the BinaryContent record for a given binarycontent_id.

Subscriptions

GET /subscriptions/
Returns a list of Subscriptions.

POST /subscriptions/
Creates a new Subscription record.

GET /subscriptions/ (**int:** *subscription_id*) **/**
Returns the Subscription record for a given subscription_id.

PUT /subscriptions/ (**int:** *subscription_id*) **/**
Updates the Subscription record for a given subscription_id.

DELETE /subscriptions/ (**int:** *subscription_id*) **/**
Deletes the Subscription record for a given subscription_id.

POST /subscriptions/ (**int:** *subscription_id*) **/send**
Triggers a send for the next Subscription message for the given subscription_id.

The actual sending is processed asynchronously by a Celery worker.

Response JSON Object

- **accepted** (*boolean*) – Whether send for subscription_id is accepted.
- **reason** (*string*) – An optional reason why the request was not accepted.

Status Codes

- 201 Created – request to send the next message accepted.
- 400 Bad Request – invalid subscription_id given.

POST /subscriptions/request
Creates a new subscription.

This endpoint is called as a webhook request from the project Hub service when a new registration is created that requires a subscription.

As such the entire payload is expected to be provided as an object in the data parameter.

Request JSON Object

- **data** (*json*) – a JSON representation of a Subscription object.

Response JSON Object

- **accepted** (*boolean*) – Whether new subscription was created.

Status Codes

- 201 Created – subscription created.
- 400 Bad Request – invalid request.

Helpers

The root URL for the helper endpoints does not include a version prefix (`https://<stage-based-messaging-store-domain>/api/`)

GET /metrics/
Returns a list of all the available metric keys provided by this service.

Status Codes

- 200 OK – no error
- 401 Unauthorized – the token is invalid/missing.

POST /metrics/
Starts a task that fires all scheduled metrics.

Status Codes

- 200 OK – no error
- 401 Unauthorized – the token is invalid/missing.

GET /health/
Returns a basic health check status.

Status Codes

- 200 OK – no error
- 401 Unauthorized – the token is invalid/missing.

CHAPTER 7

Integrations with other Seed services

The Seed Stage-based Messaging Store currently integrates with three other Seed services.

Seed Scheduler

Outgoing integrations

New subscription

When a new Subscription object is **created** locally (via the API or the webhook endpoint) an async Celery task is queued to communicate to the Seed Scheduler to create a scheduled POST-back to the Stage-based Messaging Store endpoint (`POST /subscriptions/(int:subscription_id)/send`) on the given schedule in the subscription.

Updated subscription

When a Subscription object is **updated** locally there are two integrations to the Seed Scheduler than can occur:

1. If the update is marking the Subscription as complete, an async Celery task is queued to communicate to the Seed Scheduler to deactive the scheduled POST-backs for this Subscription.
2. If the update is marking the Subscription as inactive, an async Celery task is queued to communicate to the Seed Scheduler to deactive the scheduled POST-backs for this Subscription.

Incoming integrations

Once a schedule has been setup (see *New subscription*) in the Seed Scheduler for a Subscription, the Scheduler will call the (`POST /subscriptions/(int:subscription_id)/send`) endpoint on the setup schedule.

Seed Message Sender

Outgoing integrations

Message Sending

When the Subscription send endpoint (`POST /subscriptions/(int:subscription_id)/send`) is called by the Scheduler an async Celery task is queued to process the Subscription.

During this process each message that needs to be sent will be queued by making a request to the Seed Message Sender with the relevant message and user details.

Project Hub

Incoming integrations

New registration

When a registration happens on the project hub it calls the `POST /subscriptions/request` endpoint with the subscription details to create a new Subscription for the user.

CHAPTER 8

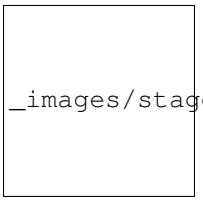
Production requirements and setup

Running in Production

The Seed Stage-Based Messaging Store is expected to be run in a Docker container and as such a Docker file is provided in the source code repository.

The web service portion and celery work portion of the Stage-Based Messaging Store are expected to be run in different instances of the same Docker container.

An example production setup might look like this:



CHAPTER 9

Indices, glossary and tables

- genindex
- modindex
- Glossary

/binarycontent

GET /binarycontent/, 15
GET /binarycontent/(int:binarycontent_id)/, 15
POST /binarycontent/, 15
PUT /binarycontent/(int:binarycontent_id)/, 15
DELETE /binarycontent/(int:binarycontent_id)/, 15

/message

GET /message/, 15
GET /message/(int:message_id)/, 15
GET /message/(int:message_id)/content/, 15
POST /message/, 15
PUT /message/(int:message_id)/, 15
DELETE /message/(int:message_id)/, 15

/messageset

GET /messageset/, 15
GET /messageset/(int:messageset_id)/, 15
GET /messageset/(int:messageset_id)/messages/, 15
POST /messageset/, 15
PUT /messageset/(int:messageset_id)/, 15
DELETE /messageset/(int:messageset_id)/, 15

/schedule

GET /schedule/, 15
GET /schedule/(int:schedule_id)/, 15
POST /schedule/, 15
PUT /schedule/(int:schedule_id)/, 15
DELETE /schedule/(int:schedule_id)/, 15

/subscriptions

GET /subscriptions/, 16
GET /subscriptions/(int:subscription_id)/, 16
POST /subscriptions/, 16
POST /subscriptions/(int:subscription_id)/send, 16
POST /subscriptions/request, 16
PUT /subscriptions/(int:subscription_id)/, 16
DELETE /subscriptions/(int:subscription_id)/, 16

/user

POST /user/token/, 14

E

environment variable
- BROKER_URL, 6
- DEBUG, 6
- IDENTITY_STORE_TOKEN, 6
- IDENTITY_STORE_URL, 6
- MESSAGE_SENDER_TOKEN, 6
- MESSAGE_SENDER_URL, 6
- METRICS_AUTH_TOKEN, 6
- METRICS_URL, 6
- SCHEDULER_API_TOKEN, 6
- SCHEDULER_INBOUND_API_TOKEN, 6
- SCHEDULER_URL, 6
- SECRET_KEY, 5
- STAGE_BASED_MESSAGING_DATABASE, 5, 6
- STAGE_BASED_MESSAGING_SENTRY_DSN, 6
- STAGE_BASED_MESSAGING_URL, 6
- USE_SSL, 6

S

STAGE_BASED_MESSAGING_DATABASE, 5