Seed Identity Store Documentation

Release 0.9

Praekelt.org

Jul 16, 2018

Contents

CHAPTER 1

Seed Identity Store documentation

The Seed Identity Store is one of the microservices in the Seed Stack.

The Identity Store has two key responsibilities:

- Store the unique Identity of any end-user created in the Seed system.
- Store the Opt-out or Opt-in status for any end-user in the Seed system.

Getting started

The following resources are provided to help you get started running or developing the Seed Identity 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.
- Learn about what is required when running the service in *Production*

CHAPTER 2

Requirements

Overview

The Seed Identity 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 Identity Store only depends on one other seed service, the Go Metrics API.

CHAPTER 3

Setup

Installing

The steps required to install the Seed Identity Service are:

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

   ```
   $ git clone https://github.com/praekelt/seed-identity-store.git
   ```

   This will create a directory `seed-identity-store` 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 Identity Store needs to exist before running this command. See *IDENTITIES_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_identity_store/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.

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

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

HOOK_AUTH_TOKEN
An Authorization Token to use when making a POST request to a webhook.

BROKER_URL
The Broker URL to use with Celery.

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

Identity

The Identity is the primary model of the Seed Identity Store. Each record represents a unique user identity.

Fields

id A UUID 4 unique identifier for the record.

version An integer number representing the schema version number that this record was created with.

details A JSON fields containing extra details of the identity record. It should always contain at least an addresses key with the following structure:

communicate_through A self-referencing link to another Identity record that should be used instead of this record when trying to communicate directly with an end-user.

operator A self-referencing link to another Identity record that represents the operator that created this record.

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.

OptIn

id A UUID 4 unique identifier for the record.

identity A reference to an Identity record.

address_type The address type used to identify the Identity.

address The address used to identify the Identity

request_source The Service that the OptIn was requested from.

requestor_source_id The ID for the user requesting the OptIn on the service that it was requested from. Ideally a UUID.

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

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

OptOut

id A UUID 4 unique identifier for the record.

identity A reference to an Identity record.

optout_type A field representing a fixed set of reasons for the OptOut:

- stop -> No communication on address
- stopall -> No communication on all addresses
- unsubscribe -> Unsubcribe
- forget -> Forget

reason An optional reason (e.g. 'not interested') for the OptOut.

address_type The address type used to identify the Identity.

address The address used to identify the Identity

request_source The Service that the OptOut was requested from.

requestor_source_id The ID for the user requesting the OptOut on the service that it was requested from. Ideally a UUID.

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

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

CHAPTER 5

Authentication and Authorization

Basics

Authentication to the Seed Identity 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: <identity-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 Identity Store provides REST like API with JSON payloads.

The root URL for all of the endpoints is:

```
https://<identity-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://sis.example.org/api/v1/enpoint/?limit=10&offset=30",
    "previous": "http://sis.example.org/api/v1/endpoint/?limit=10&offset=10",
    "results": []
}
```

Endpoints

The endpoints provided by the Seed Identity 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://<identity-store-domain>/api/v1/`)

Users and Groups

GET /user/
Returns a list of users for the Seed Identity Store service.

Status Codes

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

Example request:

```
GET /user/ HTTP/1.1
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Example response:

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

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "email": "john@example.org",
            "groups": [],
            "url": "http://sis.example.org/api/v1/user/1/",
            "username": "john"
        }
    ]
}
```

GET /user/ (**int:** *user_id*) **/**
Returns the details of the specified user ID.

Parameters

- **user_id** (*int*) – a user's unique ID.

Status Codes

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

Example request:

```
GET /user/1/ HTTP/1.1
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Example response:

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

{
    "email": "john@example.org",
    "groups": [],
    "url": "http://sis.example.org/api/v1/user/1/",
    "username": "john"
}
```

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"
}
```

GET /group/

Returns a list of groups for the Seed Identity Store service.

Status Codes

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

Example request:

```
GET /group/ HTTP/1.1
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Example response:

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

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "name": "Admins",
            "url": "http://sis.example.org/api/v1/group/1/"
        }
    ]
}
```

GET /group/ (**int:** *group_id*) **/**

Returns the details of the specified group ID.

Parameters

- **group_id** (*int*) – a group's unique ID.

Status Codes

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

Example request:

```
GET /group/1/ HTTP/1.1
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Example response:

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

{
    "name": "Admins",
    "url": "http://sis.example.org/api/v1/group/1/"
}
```

Identities

GET /identities/
Returns a list of identities.

Status Codes

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

Example request:

```
GET /identities/ HTTP/1.1
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Example response:

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

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
    ]
}
```

POST /identities/
Create a new identity.

Request JSON Object

- **version** (*int*) – optional version number of the Identity schema being used.
- **details** (*json*) – a JSON object representing the Identity details
- **communicate_through** (*string*) – optional URL to another identity that represents an Identity to communicate through.
- **operator** (*string*) – optional optional URL to another identity that presents an operator that is responsible for this Identity

Response JSON Object

- **id** (*uuid*) – the UUID of this Identity
- **version** (*int*) – the version number of the Identity schema being used.
- **details** (*json*) – the JSON object representing the Identity details
- **communicate_through** (*url*) – URL to another identity that represents an Identity to communicate through.
- **operator** (*url*) – URL to another identity that presents an operator that is responsible for this Identity
- **created_at** (*datetime*) – the date and time this Identity was created
- **created_by** (*datetime*) – the ID of the user that created this Identity
- **updated_at** (*datetime*) – the date and time this Identity was last updated
- **updated_by** (*datetime*) – the ID of the user that created this Identity

Status Codes

- 201 Created – identity successfully created.
- 400 Bad Request – the details field was not provided or was invalid.
- 401 Unauthorized – the token is invalid/missing.

Example request:

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

{
    "details": {
        "addresses": {
        "msisdn": {
            "+27115551234": {}
        },
    },
    "version": 1
}
```

Example response:

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

{
    "communicate_through": null,
    "created_at": "2016-09-30T11:10:21.693326Z",
    "created_by": 1,
    "details": {
        "addresses": {
           "msisdn": {
                "+27115551234": {}
            }
        }
    }
}
    },
    "id": "4be7c1f9-f3a1-4bb3-ade7-a193ca2e79d0",
    "operator": null,
    "updated_at": "2016-09-30T11:10:21.693364Z",
    "updated_by": 1,
    "version": 1
}
```

GET /identities/ (**uuid:** *identity_id*) **/**
Returns the Identity record for a given UUID.

Status Codes

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

PUT /identities/ (**uuid:** *identity_id*) **/**
Update the Identity record for the given UUID.

Status Codes

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

DELETE /identities/ (**uuid:** *identity_id*) **/**
Delete the Identity record for the given UUID.

Status Codes

- 204 No Content – delete successfully completed.
- 401 Unauthorized – the token is invalid/missing.

GET /identities/ (**uuid:** *identity_id*) **/addresses/**
str: *address_type***/** Searches address by a given type for a given Identity.

Status Codes

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

GET /identities/search/
Search Identity records by specifying Django filter keys as query parameters.

Status Codes

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

POST /optout/
Create an OptOut record.

Status Codes

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

POST /optin/
Create an OptIn record.

Status Codes

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

Other

GET /detailkeys/
Returns a list of all the unique keys stored in any *detail* field of an Identity record.

This list is populated by a post-save signal on the Identity record.

Status Codes

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

GET /webhook/
Returns a list of setup webhooks.

Status Codes

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

POST /webhook/
Creates a webhook.

Status Codes

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

GET /webhook/ (**int:** *webhook_id*) **/**
Get the details of a webhook specified by webhook_id.

Status Codes

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

PUT /webhook/ (**int:** *webhook_id*) **/**
Updates the details of a webhook specified by webhook_id.

Status Codes

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

DELETE /webhook/ (**int:** *webhook_id*) **/**
Deletes the webhook specified by webhook_id.

Status Codes

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

Helpers

The root URL for the helper endpoints does not include a version prefix (`https://<identity-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

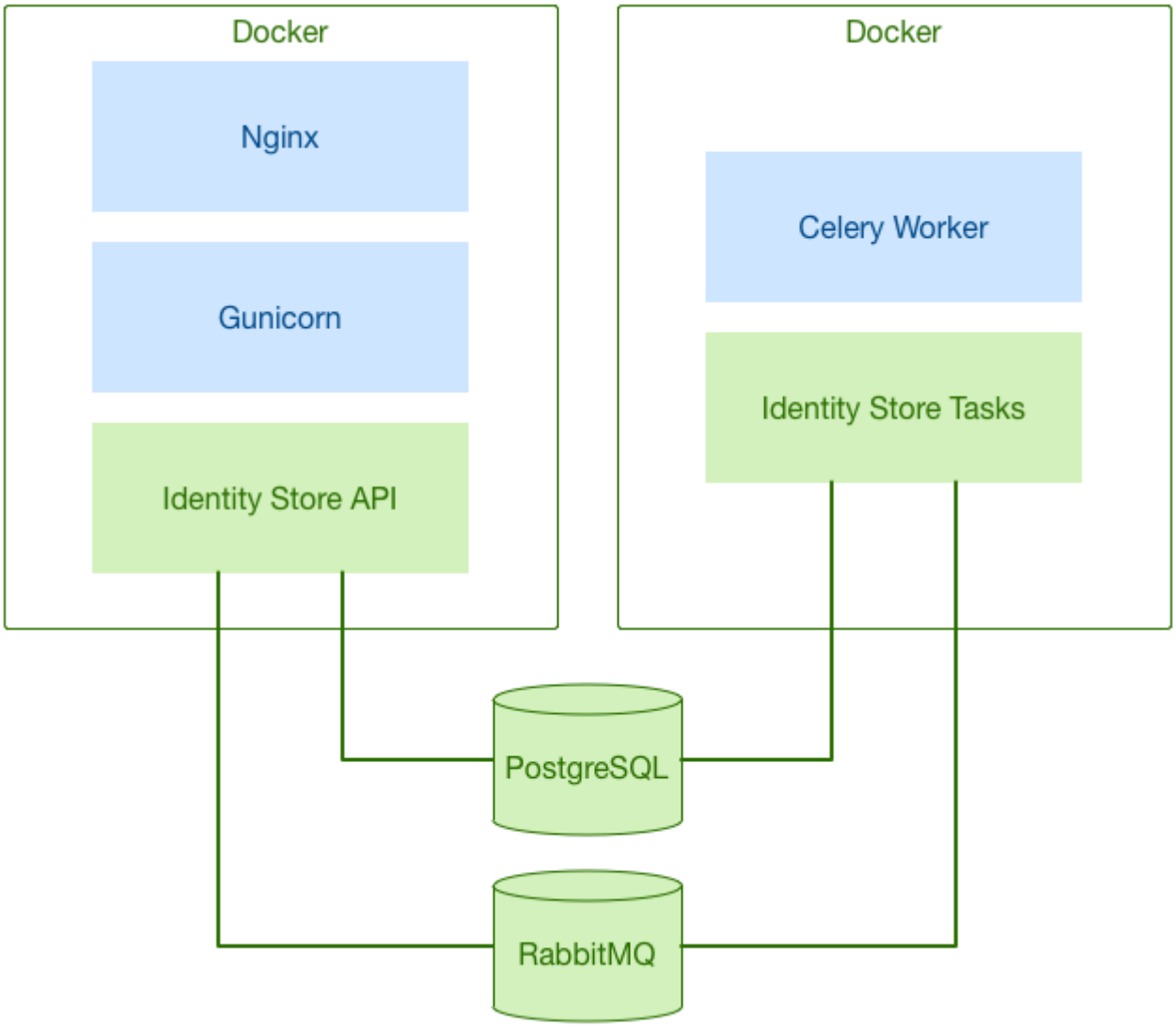
Production requirements and setup

Running in Production

The Seed Identity 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 Identity Store are expected to be run in different instances of the same Docker container.

An example production setup might look like this:


Docker
Nginx
Gunicorn
Identity Store API
Docker
Celery Worker
Identity Store Tasks
PostgreSQL
RabbitMQ

CHAPTER 8

Indices, glossary and tables

- genindex
- modindex
- Glossary

/detailkeys

/group

/identities

/optin

/optout

/user

/webhook

E

I