CritiqueBrainz Documentation

Release 0.1

MetaBrainz Foundation

April 01, 2016

Contents

Hello there! Thanks for your interest in CritiqueBrainz project. It is a repository for Creative Commons licensed music reviews. You see what other people have written about an album or write your own review! This project is based on data from MusicBrainz - open music encyclopedia. Everyone - including you - can participate and contribute.

This is an open source project. Source code is available on GitHub.

Documentation is divided into two parts:

1. Information for developers who want to use our web service to get reviews and do some fun stuff with them.
2. Information for contributors, who want to make CritiqueBrainz a better project. You can help by writing code, translating the interface, or doing something else.

CHAPTER 1

Contents

1.1 Development

This part of CritqueBrainz documentation will help you find information about our web service.

1.1.1 Server Web API

CritiqueBrainz provides various endpoints that can be used to interact with the data. Web API uses JSON format.

Root URL: `https://critiquebrainz.org/ws/1`

Reference

Below you will find description of all available endpoints.

Reviews

GET /review/languages

Get list of supported review languages (language codes from ISO 639-1).

Response Headers

- Content-Type – *application/json*

GET /review/

Get list of reviews.

JSON Parameters

- **entity_id** (*uuid*) – UUID of the release group that is being reviewed
- **entity_type** (*string*) – One of the supported reviewable entities. 'release_group' or 'event' etc. **(optional)**

Query Parameters

- **user_id** – user's UUID **(optional)**
- **sort** – `rating` or `created` **(optional)**
- **limit** – results limit, min is 0, max is 50, default is 50 **(optional)**
- **offset** – result offset, default is 0 **(optional)**

- **`language`** – language code (ISO 639-1) **(optional)**

Response Headers

- Content-Type – *application/json*

`POST /review/`

Publish a review.

OAuth scope: review

Request Headers

- Content-Type – *application/json*

JSON Parameters

- **`entity_id`** (*uuid*) – UUID of the release group that is being reviewed
- **`entity_type`** (*string*) – One of the supported reviewable entities. 'release_group' or 'event' etc.
- **`text`** (*string*) – review contents, min length is 25, max is 5000
- **`license_choice`** (*string*) – license ID
- **`lang`** (*string*) – language code (ISO 639-1), default is `en` **(optional)**
- **`is_draft`** (*boolean*) – whether the review should be saved as a draft or not, default is `False` **(optional)**

Response Headers

- Content-Type – *application/json*

`GET /review/`(**uuid:** *review_id*)**`/revisions/`**

int: *rev* Get a particular revisions of review with a specified UUID.

Status Codes

- 200 OK – no error
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

`GET /review/`(**uuid:** *review_id*)**`/revisions`**

Get revisions of review with a specified UUID.

Status Codes

- 200 OK – no error
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

`POST /review/`(**uuid:** *review_id*)**`/report`**

Create spam report for a specified review.

OAuth scope: vote

Response Headers

- Content-Type – *application/json*

`GET /review/`(**uuid:** *review_id*)**`/vote`**
Get your vote for a specified review.

OAuth scope: vote

Response Headers

- Content-Type – *application/json*

`PUT /review/`(**uuid:** *review_id*)**`/vote`**
Set your vote for a specified review.

OAuth scope: vote

JSON Parameters

- **`vote`** (*boolean*) – `true` if upvote, `false` if downvote

Status Codes

- 200 OK – success
- 400 Bad Request – invalid request (see source)
- 403 Forbidden – daily vote limit exceeded
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

`DELETE /review/`(**uuid:** *review_id*)**`/vote`**
Delete your vote for a specified review.

OAuth scope: vote

Response Headers

- Content-Type – *application/json*

`GET /review/`(**uuid:** *review_id*)
Get review with a specified UUID.

Status Codes

- 200 OK – no error
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

`DELETE /review/`(**uuid:** *review_id*)
Delete review with a specified UUID.

OAuth scope: review

Status Codes

- 200 OK – success
- 403 Forbidden – access denied
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

POST /review/ (**uuid:** *review_id*)
Update review with a specified UUID.

OAuth scope: review

Status Codes

- 200 OK – success
- 403 Forbidden – access denied
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

Users

GET /user/me/applications
Get your applications.

Response Headers

- Content-Type – *application/json*

GET /user/me/reviews
Get your reviews.

Response Headers

- Content-Type – *application/json*

GET /user/me/tokens
Get your OAuth tokens.

Response Headers

- Content-Type – *application/json*

GET /user/me
Get your profile information.

Query Parameters

- **inc** – includes

Response Headers

- Content-Type – *application/json*

POST /user/me
Modify your profile.

OAuth scope: user

Request Headers

- Content-Type – *application/json*

JSON Parameters

- **display_name** (*string*) – Display name **(optional)**
- **email** (*string*) – Email address **(optional)**
- **show_gravatar** (*boolean*) – Show gravatar **(optional)**

Response Headers

- Content-Type – *application/json*

`DELETE /user/me`

Delete your profile.

OAuth scope: user

Response Headers

- Content-Type – *application/json*

`GET /user/`

Get list of users.

Query Parameters

- **`limit`** – results limit, min is 0, max is 50, default is 50 **(optional)**
- **`offset`** – result offset, default is 0 **(optional)**

Response Headers

- Content-Type – *application/json*

`GET /user/`(**uuid:** *user_id*)

Get profile of a user with a specified UUID.

Response Headers

- Content-Type – *application/json*

OAuth

`POST /oauth/token`

OAuth 2.0 token endpoint.

Form Parameters

- **`string client_id`** –
- **`string client_secret`** –
- **`string redirect_uri`** –
- **`string grant_type`** – `authorization_code` or `refresh_token`
- **`string code`** – (not required if grant_type is `refresh_token`)
- **`string refresh_token`** – (not required if grant_type is `authorization_code`)

Response Headers

- Content-Type – *application/json*

1.1.2 OAuth 2.0

Introduction

OAuth 2.0 is a protocol that lets you create applications that can request access different parts of user profiles. This page describes how to use OAuth 2.0 when accessing a server API from a web server application.

All developers need to register their application before getting started. A registered OAuth application is assigned a unique Client ID and Client Secret. The Client Secret should not be shared.

Authorization

Requesting authorization

The authorization process starts by redirecting the user to the authorization endpoint with a set of query string parameters describing the authorization request. The endpoint is located at `https://critiquebrainz.org/oauth/authorize` and is only available over HTTPS. HTTP connections are refused.

Parameters:

Parameter	Description
response_type	Desired grant type. Must be set to `code`.
client_id	Client ID assigned to your application.
redirect_uri	URL where clients should be redirected after authorization. It must match exactly the URL you entered when registering your application.
scope	*Optional.* Comma separated set of scopes. Identifies resources that your application will have access to. You should request only the scopes that your application needs. See *Scopes* for more info.
state	*Optional.* Random string that is used to protect against cross-site request forgery attacks. Server roundtrips this parameter, so your application receives the same value it sent.

Example:

```
https://critiquebrainz.org/oauth/authorize?
   response_type=code&
   scope=review,vote&
   redirect_uri=http%3A%2F%2Fwww.example.com.com%2Fcallback&
   client_id=yDDvwAzPUnoD8imvTpVm&
   access_type=offline
```

Handling server response

The response will be sent to the `redirect_url` specified in configuration of your application. If user approves access request, then the response will contain authorization code and `state` parameter (if it was included in the request):

```
https://www.example.com/callback?state=a35Bsw1koA3pM34&code=31Uq7v15Qqm9g8YcoUT31D
```

If the user does not approve the request, the response will contain an error message:

```
https://www.example.com/callback?state=a35Bsw1koA3pM34&error=access_denied
```

Access token

Exchanging authorization code for an access token

After your application receives an authorization code, it can send a *POST* request the token endpoint located at `https://critiquebrainz.org/ws/1/oauth/token`, to exchange the authorization code for an access token. As before, this endpoint is only available over HTTPS and HTTP requests will be refused. The request includes following parameters:

Parameter	Description
code	The authorization code returned from authorization request.
client_id	Client ID assigned to your application.
client_secret	Client secret assigned to your application.
redirect_uri	URL where response will be sent. Must match your application configuration.
grant_type	Must be set to `authorization_code`.

Token request might look like this:

```
POST /ws/1/oauth/token HTTP/1.1
Host: critiquebrainz.org
Content-Type: application/x-www-form-urlencoded

code=3lUq7v15Qqm9g8YcoUT31D&
client_id=yDDvwAzPUnoD8imvTpVm&
client_secret=AFjfpM7Ar1KtD0bnfV5InQ&
redirect_uri=http%3A%2F%2Fwww.example.com%2Fcallback&
grant_type=authorization_code
```

Successful response to this request will contain the following fields:

Field	Description
access_token	Access token that can be used to get data from CritiqueBrainz API
token_type	Type of returned access token. Will have `Bearer` value.
expires_in	The remaining lifetime of returned access token.
refresh_token	Token that can be used to obtain new access token. See below for more info about this.

Response example:

```
{
  "access_token": "2YotnFZFEjr1zCsicMWpAA",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA"
}
```

Refreshing an access token

To obtain a new access token, your application needs to send POST request to `https://critiquebrainz.org/ws/1/oauth/token`. The request must include the following parameters:

Parameter	Description
refresh_token	The refresh token returned during the authorization code exchange.
client_id	Client ID assigned to your application.
client_secret	Client secret assigned to your application.
redirect_uri	URL where response will be sent. Must match your application configuration.
grant_type	Must be set to `refresh_token`

Request might look like this:

```
POST /ws/1/oauth/token HTTP/1.1
Host: critiquebrainz.org
Content-Type: application/x-www-form-urlencoded

refresh_token=tGzv3JOkF0XG5Qx2TlKWIA&
```

```
client_id=yDDvwAzPUnoD8imvTpVm&
client_secret=AFjfpM7Ar1KtD0bnfV5InQ&
redirect_uri=http%3A%2F%2Fwww.example.com%2Fcallback&
grant_type=refresh_token
```

As long as the user has not revoked the access granted to your application, you will receive response that will look like this:

```
{
  "access_token": "zIYanFZFEjr1zCsicMWpo6",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "PUnoD8im10XG5QxGzv3JO1"
}
```

Scopes

Authorization requests have a limited scope. You should request only the scopes that your application necessarily needs. CritiqueBrainz provides the following scopes:

- `review` - Create and modify reviews.
- `vote` - Submit and delete votes on reviews.
- `user` - Modify profile info and delete profile.

1.2 Contributing

1.2.1 Introduction

CritiqueBrainz project is separated into three main packages: data, frontend, and web service (ws). The data package is used to interact with the database. The frontend provides user-friendly interface that is available at https://critiquebrainz.org. The web service provides web API for CritiqueBrainz (see Server Web API).

Here's an overview of the project structure:

Contributing

See CONTRIBUTING.md file.

Using Vagrant

Vagrant significantly simplifies development process on all major platforms by running applications in reproducible environment. It is available at http://www.vagrantup.com/.

You can use it for CritiqueBrainz development. All you need to do is set up custom configuration file. After that you can start a VM and connect to it:

```
$ vagrant up
$ vagrant ssh
```

After VM is created and running, you can start the application:

```
$ cd /vagrant
$ python run.py
```

Web server should be accessible at http://localhost:8080/.

PostgreSQL will also be available on port *15432* with trust authentication method.

Testing

To run all tests use:

```
$ fab test
```

This command run all tests and, if successful, produce a test coverage report.

Modifying strings

CritiqueBrainz supports interface translation. If you add or modify strings that will be displayed to users, then you need to wrap them in one of two functions: `gettext()` or `ngettext()`.

Before committing changes don't forget to extract all strings into `messages.pot`:

> $ fab update_strings

For more info see Translation.

1.2.2 Installation

Requirements

- Python (tested on 2.7.4)
- `python-dev`
- PostgreSQL (tested on 9.1.9)
- `postgresql-contrib`
- `postgresql-server-dev-9.1`
- virtualenv
- memcached
- git
- nodejs & npm
- lessc

How to start

Creating virtualenv (optional)

This step is not required, but highly recommended for development instances. You may find it useful to keep libraries needed by CritiqueBrainz seperated from your global python libraries. To achieve this, you will need a `virtualenv` package. You may install it with `pip` or `apt-get` on Debian/Ubuntu systems:

```
$ sudo pip install virtualenv (or)
$ sudo apt-get install python-virtualenv
```

Then run:

```
$ scripts/admin.sh
```

It will create a symbolic link `env` to virtualenv's entry point in your `critiquebrainz/` directory. To enter newly created virtualenv, type in:

```
$ source ./env
```

You should do this before executing any other file from CritiqueBrainz package.

Installing dependencies

If you're in your desired python environment, simply run:

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

to install all required dependencies.

Configuration

First, you need to create custom configuration file. Copy the skeleton configuration:

```
$ cp critiquebrainz/config.py.example critiquebrainz/config.py
```

Then, open `critiquebrainz/config.py` in your favourite text editor, uncomment `SQLALCHEMY_DATABASE_URI` variable, and fill in the fields in angle brackets.

Database initialization

Now, you need to create and configure the database with:

```
$ python manage.py create_db
```

This command will

- create new PostgreSQL role, if needed
- create new PostgreSQL database, if needed
- register `uuid-ossp` PostgreSQL extension, if needed

You also need to add initial data for models (predefined licenses). To do this use `fixtures` command:

```
$ python manage.py fixtures
```

Importing data We provide daily data dumps from https://critiquebrainz.org that include reviews and most of the data associated with them. If you want to import that into your own installation, download archives from ftp://ftp.musicbrainz.org/pub/musicbrainz/critiquebrainz/dump/ (you'll need to get the base archive `cbdump.tar.bz2` and one with reviews) and use `python manage.py export importer` command. First you need to import base archive and then one that contains reviews. For example:

```
$ python manage.py export importer cbdump.tar.bz2
$ python manage.py export importer cbdump-reviews-all.tar.bz2
```

Keep in mind that CritiqueBrainz only supports importing into an empty database.

Preparing login

Before you begin using authentication with MusicBrainz accounts, you need to set `MUSICBRAINZ_CLIENT_ID` and `MUSICBRAINZ_CLIENT_SECRET` in `critiquebrainz/config.py`. To obtain these keys, you need to register your instance of CrituqeBrainz on MusicBrainz.

Note `<your domain>` field in the urls listed below should probably be set to `127.0.0.1:8080`, if you plan to run your CritiqueBrainz instance locally in development mode.

MusicBrainz You need MusicBrainz account to register your application. Then head to https://musicbrainz.org/account/applications/register and follow the instructions. In `Callback URL` field type:

```
http://<your domain>/login/musicbrainz/post
```

Finally, save the obtained `OAuth Client ID` and `OAuth Client Secret` fields in your `config.py` fields `MUSICBRAINZ_CLIENT_ID` and `MUSICBRAINZ_CLIENT_SECRET` respectively.

Security

Unless you are doing development, it might be a good idea to make your installation HTTPS-only. You can read about pros and cons at https://security.stackexchange.com/questions/258/. If you don't want to do that yet, here's a list of blueprints that should be kept secure:

- `profile_details` (private user info)
- `profile_applications` (secret info about user's applications)
- `ws_oauth` (OAuth 2.0 token endpoint)

More information about importance of keeping transport layer secure is available at https://www.owasp.org/index.php/Top_10_2010-A9-Insufficient_Transport_Layer_Protection.

Compiling Stylesheets

CritiqueBrainz stylesheets are written in *less*. To compile them to *css* you need to have nodejs installed:

```
$ curl -sL https://deb.nodesource.com/setup | sudo bash -
$ sudo apt-get install -y nodejs
```

Once node is installed, you can install the less compiler:

```
$ npm install -g less less-plugin-clean-css
```

Now, to actually compile the stylesheets:

```
$ fab compile_styling
```

You can also compile the stylesheets directly in your browser by setting the `COMPILE_LESS` field to `True` in your `config.py`.

Running the server

To run the server you can use `run.py` script:

```
$ python run.py
```

1.2.3 Database

CritiqueBrainz uses PostgresSQL DBMS to save reviews and other data.

Entities

- Review
- Revision
- License
- User
- Vote
- SpamReport
- OAuthClient
- OAuthGrant
- OAuthToken

Schema diagram

1.2.4 Exporting data

You can create backups including various pieces of data that we store: reviews, revisions, users, and other stuff. Some parts include private data about users that is not meant to be shared.

Creating data dumps

Below you can find commands that can be used to create backups of different formats.

Complete database dump *(for PostgreSQL)*:

```
$ python manage.py export full_db
```

MusicBrainz-style dump public *(no private info)*:

```
$ python manage.py export public
```

JSON dump with all reviews *(no private info)*:

```
$ python manage.py export json
```

All commands have rotation feature which can be enabled by passing *-r* argument.

1.2.5 Translation

Our goal is to make CritiqueBrainz project accessible to readers and writers all over the world. You can post reviews in any language you want, but it is also important to provide good way to find these reviews. That's why we need your help!

Translation process is being done on Transifex platform. You can submit your translation suggestions there and collaborate with other translators. If you want to contribute translations, that's where you should to do it.

Our project page is located at https://www.transifex.com/projects/p/critiquebrainz/.

There are a couple of things you should know if you are trying to modify strings. See the info below.

Extracting strings

To extract strings into a Portable Object Template file (*messages.pot*) use command:

```
$ fab extract_strings
```

Adding support for a new language

To add support for new language add its Transifex code into `SUPPORTED_LANGUAGES` list in default configuration file: `default_config.py`. After that you can pull translations from Transifex:

```
$ fab pull_translations
```

You will need to create *.transifexrc* file that will look like:

```
[https://www.transifex.com]
hostname = https://www.transifex.com
username = <YOUR_EMAIL>
password = <YOUR_PASSWORD>
token =
```

More info about setup process is available at http://docs.transifex.com/developer/client/setup.

Additional info

CritiqueBrainz uses Flask-Babel extension. Its documentation is available at http://pythonhosted.org/Flask-Babel/.