musicbrainzngs Documentation

Release 0.3

Alastair Porter et. al

October 08, 2013

CONTENTS

CHAPTER

ONE

INSTALLATION

1.1 Package manager

If you want the latest stable version of musicbrainzngs, the first place to check is your systems package manager. Being a relatively new library, you might not be able to find it packaged by your distribution and need to use one of the alternate installation methods.

1.2 PyPI

Musicbrainzngs is available on the Python Package Index. This makes installing it with pip as easy as:

```
pip install musicbrainzngs
```

1.3 Git

If you want the latest code or even feel like contributing, the code is available on Github.

You can easily clone the code with git:

```
git clone git://github.com/alastair/python-musicbrainz-ngs.git
```

Now you can start hacking on the code or install it system-wide:

```
python setup.py install
```

CHAPTER

TWO

USAGE

2.1 Identification

To access the MusicBrainz webservice through this library, you need to identify your application by setting the useragent header made in HTTP requests to one that is unique to your application.

To ease this, the convenience function `musicbrainzngs.set_useragent()` is provided which automatically sets the useragent based on information about the application name, version and contact information to the format recommended by MusicBrainz.

If a request is made without setting the useragent beforehand, a `musicbrainzngs.UsageError` will be raised.

2.2 Authentication

Certain calls to the webservice require user authentication prior to the call itself. The affected functions state this requirement in their documentation. The user and password used for authentication are the same as for the MusicBrainz website itself and can be set with the `musicbrainzngs.auth()` method. After calling this function, the credentials will be saved and automaticall used by all functions requiring them.

If a method requiring authentication is called without authenticating, a `musicbrainzngs.UsageError` will be raised.

If the credentials provided are wrong and the server returns a status code of 401, a `musicbrainzngs.AuthenticationError` will be raised.

2.3 Getting data

2.4 Searching

2.5 Browsing

2.6 Submitting

CHAPTER

THREE

API

3.1 General

musicbrainzngs.**auth**(*u*, *p*)

Set the username and password to be used in subsequent queries to the MusicBrainz XML API that require authentication.

musicbrainzngs.**set_rate_limit**(*limit_or_interval=1.0*, *new_requests=1*)

Sets the rate limiting behavior of the module. Must be invoked before the first Web service call. If the *limit_or_interval* parameter is set to False then rate limiting will be disabled. If it is a number then only a set number of requests (*new_requests*) will be made per given interval (*limit_or_interval*).

musicbrainzngs.**set_useragent**(*app*, *version*, *contact=None*)

Set the User-Agent to be used for requests to the MusicBrainz webservice. This must be set before requests are made.

musicbrainzngs.**set_hostname**(*new_hostname*)

Set the base hostname for MusicBrainz webservice requests. Defaults to 'musicbrainz.org'.

3.2 Getting data

musicbrainzngs.**get_artist_by_id**(*id*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_label_by_id**(*id*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_recording_by_id**(*id*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_recordings_by_echoprint**(*echoprint*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_recordings_by_puid**(*puid*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_recordings_by_isrc**(*isrc*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_release_group_by_id**(*id*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_release_by_id**(*id*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

musicbrainzngs.**get_releases_by_discid**(*id*, *includes=[ ]*, *release_status=[ ]*, *release_type=[ ]*)

`musicbrainzngs.`**`get_work_by_id`**(*id*, *includes=[ ]*)

`musicbrainzngs.`**`get_works_by_iswc`**(*iswc*, *includes=[ ]*)

`musicbrainzngs.`**`get_collections`**()

`musicbrainzngs.`**`get_releases_in_collection`**(*collection*)

3.3 Searching

`musicbrainzngs.`**`search_annotations`**(*query='‘*, *limit=None*, *offset=None*, *strict=False*, ***fields*)
Search for annotations by a free-form *query* string or any of the following keyword arguments specifying field queries: entity, name, text, type When *fields* are set, special lucene characters are escaped in the *query*.

`musicbrainzngs.`**`search_artists`**(*query='‘*, *limit=None*, *offset=None*, *strict=False*, ***fields*)
Search for artists by a free-form *query* string or any of the following keyword arguments specifying field queries: arid, artist, sortname, type, begin, end, comment, alias, country, gender, tag When *fields* are set, special lucene characters are escaped in the *query*.

`musicbrainzngs.`**`search_labels`**(*query='‘*, *limit=None*, *offset=None*, *strict=False*, ***fields*)
Search for labels by a free-form *query* string or any of the following keyword arguments specifying field queries: laid, label, sortname, type, code, country, begin, end, comment, alias, tag When *fields* are set, special lucene characters are escaped in the *query*.

`musicbrainzngs.`**`search_recordings`**(*query='‘*, *limit=None*, *offset=None*, *strict=False*, ***fields*)
Search for recordings by a free-form *query* string or any of the following keyword arguments specifying field queries: rid, recording, isrc, arid, artist, artistname, creditname, reid, release, type, status, tracks, tracksrelease, dur, qdur, tnum, position, tag When *fields* are set, special lucene characters are escaped in the *query*.

`musicbrainzngs.`**`search_release_groups`**(*query='‘*, *limit=None*, *offset=None*, *strict=False*, ***fields*)
Search for release groups by a free-form *query* string or any of the following keyword arguments specifying field queries: rgid, releasegroup, reid, release, arid, artist, artistname, creditname, type, tag When *fields* are set, special lucene characters are escaped in the *query*.

`musicbrainzngs.`**`search_releases`**(*query='‘*, *limit=None*, *offset=None*, *strict=False*, ***fields*)
Search for releases by a free-form *query* string or any of the following keyword arguments specifying field queries: reid, release, arid, artist, artistname, creditname, type, status, tracks, tracksmedium, discids, discidsmedium, mediums, date, asin, lang, script, country, date, label, catno, barcode, puid When *fields* are set, special lucene characters are escaped in the *query*.

`musicbrainz.`**`VALID_INCLUDES`** **= {‘echoprint’: [’artists’, ‘releases’], ‘collection’: [’releases’], ‘isrc’: [’artists’, ‘releases’, ‘p**

`musicbrainz.`**`VALID_SEARCH_FIELDS`** **= {‘artist’: [’arid’, ‘artist’, ‘sortname’, ‘type’, ‘begin’, ‘end’, ‘comment’, ‘alias’, ‘co**

3.4 Browsing

`musicbrainzngs.`**`browse_artists`**(*recording=None*, *release=None*, *release_group=None*, *includes=[ ]*, *limit=None*, *offset=None*)

`musicbrainzngs.`**`browse_labels`**(*release=None*, *includes=[ ]*, *limit=None*, *offset=None*)

`musicbrainzngs.`**`browse_recordings`**(*artist=None*, *release=None*, *includes=[ ]*, *limit=None*, *offset=None*)

`musicbrainzngs.`**`browse_release_groups`**(*artist=None*, *release=None*, *release_type=[ ]*, *includes=[ ]*, *limit=None*, *offset=None*) musicbrainzngs.**browse_releases**(*artist=None*, *label=None*, *recording=None*, *release_group=None*, *release_status=[]*, *release_type=[]*, *includes=[]*, *limit=None*, *offset=None*)

3.5 Submitting

musicbrainzngs.**submit_barcodes**(*barcodes*)

Submits a set of {release1: barcode1, release2:barcode2}

Must call auth(user, pass) first

musicbrainzngs.**submit_puids**(*puids*)

Submit PUIDs.

Must call auth(user, pass) first

musicbrainzngs.**submit_echoprints**(*echoprints*)

Submit echoprints.

Must call auth(user, pass) first

musicbrainzngs.**submit_isrcs**(*recordings_isrcs*)

Submit ISRCs. Submits a set of {recording-id: [isrc1, isrc2, ...]}

Must call auth(user, pass) first

musicbrainzngs.**submit_tags**(*artist_tags={}*, *recording_tags={}*)

Submit user tags. Artist or recording parameters are of the form: {'entityid': [taglist]}

Must call auth(user, pass) first

musicbrainzngs.**submit_ratings**(*artist_ratings={}*, *recording_ratings={}*)

Submit user ratings. Artist or recording parameters are of the form: {'entityid': rating}

Must call auth(user, pass) first

musicbrainzngs.**add_releases_to_collection**(*collection*, *releases=[]*)

Add releases to a collection. Collection and releases should be identified by their MBIDs

Must call auth(user, pass) first

musicbrainzngs.**remove_releases_from_collection**(*collection*, *releases=[]*)

Remove releases from a collection. Collection and releases should be identified by their MBIDs

Must call auth(user, pass) first

3.6 Exceptions

class musicbrainzngs.**AuthenticationError**(*message=None*, *cause=None*)

Received a HTTP 401 response while accessing a protected resource.

class musicbrainzngs.**UsageError**

Error related to misuse of the module API.

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *search*