
python-pinballmap Documentation

Release 0.3.4

Isaac Csandl

Jan 13, 2019

Contents:

1	Pinball Map API Client	1
1.1	Installation	1
1.2	Quick Start	1
1.3	Command Line Usage	2
1.4	Example Django <code>settings.py</code>	4
1.5	Example Django management command	4
2	Change Log	5
2.1	0.3.4	5
2.2	0.3.3	5
2.3	0.2.0	5
2.4	0.1.2	6
3	Roadmap	7
4	PinballMapClient docs	9
5	Indices and tables	13
	Python Module Index	15

CHAPTER 1

Pinball Map API Client

Python client for [Pinball Map API](#).

Special thanks to [Logan Arcade](#) in Chicago, IL for supporting development of this project.

Current version: 0.3.4

[Source on GitHub](#)

[pinballmap](#) at [Python Package Index \(PyPI\)](#)

[Documentation at Read the Docs](#)

1.1 Installation

```
$ pip install pinballmap
```

1.2 Quick Start

```
>>> from pinballmap import PinballMapClient

>>> c = PinballMapClient(location_id=0, region_name="chicago", authentication_token=".",
↳ user_email="email@example.com")

# Sync your list of machines by providing a complete list of current Pinball Map_
↳ machine_ids.
# e.g.:
>>> c.update_map([1423, 22, 33, 44, 423, 55])

# look up games by name, results sorted by match quality:
>>> c.machine_by_name("Game of Thrones (LE)")
({'created_at': '2015-10-22T18:55:02.702Z',
```

(continues on next page)

(continued from previous page)

```
'id': 2442,
'ipdb_id': None,
'ipdb_link': '',
'is_active': None,
'machine_group_id': 22,
'manufacturer': 'Stern',
'name': 'Game of Thrones (LE)',
'updated_at': '2015-10-22T18:55:02.702Z',
'year': 2015}, ...)
```

1.3 Command Line Usage

Limited functionality at this point, but it does a few things.

```
>>> pinballmap --help

usage: pinballmap [-h] [-l LOCATION_ID] [-r REGION_NAME] [-i]
                  [-t AUTHENTICATION_TOKEN] [-e USER_EMAIL] [-p USER_PASSWORD]
                  {search,machine_id,machine_ipdb,loc_machines,get_token}
                  [value [value ...]]

Interact with the Pinball Map API from the command line.

positional arguments:
  {search,machine_id,machine_ipdb,loc_machines,get_token}
    search: finds machine data by name; machine_id: finds
    machine matching id; machine_ipdb: finds machine by
    IPDB id; loc_machines: list machines at a location
    get_token: get an auth token for email and password
  value

optional arguments:
  -h, --help            show this help message and exit
  -l LOCATION_ID, --location LOCATION_ID
                        region name (e.g., chicago)
  -r REGION_NAME, --region REGION_NAME
                        region name (e.g., chicago)
  -i, --id-only          return only machine ids for query
  -t AUTHENTICATION_TOKEN, --token AUTHENTICATION_TOKEN
                        API authentication token (needed for all write
                        operations)
  -e USER_EMAIL, --email USER_EMAIL
                        API User email address (required for all write
                        operations)
  -p USER_PASSWORD, --password USER_PASSWORD
                        API User password (required if you are not providing a
                        token with -t/--token)

Happy flipping! This is python-pinballmap v0.2.2, supporting Pinball Map API
v1.0

>>> pinballmap search 'Game of Thrones (LE)'
```

(continues on next page)

(continued from previous page)

id	name	manufacturer	year	ipdb_id
2442	Game of Thrones (LE)	Stern	2015	
2441	Game of Thrones (Pro)	Stern	2015	
2527	Game of Thrones (Premium)	Stern	2015	
760	The Bally Game Show	Bally	1990	985


```
>>> pinballmap --location 4495 loc_machines
```

id	name	manufacturer	year	ipdb_id
1296	AC/DC (Premium)	Stern	2012	5775
2832	Attack From Mars (Remake)	Chicago Gaming	2017	
2728	Batman 66 (LE)	Stern	2016	6355
3022	Deadpool (Pro)	Stern	2018	
738	Dr. Dude	Bally	1990	737
2442	Game of Thrones (LE)	Stern	2015	6309
2571	Ghostbusters (LE)	Stern	2016	6334
2875	Guardians of the Galaxy (Pro)	Stern	2017	6474
2924	Iron Maiden: Legacy of the Beast (Premium)	Stern	2018	
695	Junk Yard	Williams	1996	4014
2353	Kiss	Stern	2015	6265
2306	Medieval Madness (Remake)	Chicago Gaming	2015	6263
1606	Metallica (Premium)	Stern	2013	6030
641	Monster Bash	Williams	1998	4441
723	PIN-BOT	Williams	1986	1796
677	Radical!	Bally	1990	1904
1276	Shaq Attaq	Gottlieb	1995	2874
2565	Spider-Man (Vault Edition)	Stern	2016	6328
684	Star Trek: The Next Generation	Williams	1993	2357
694	Star Wars	Data East	1992	2358
2844	Star Wars (Premium)	Stern	2017	6429
1118	TRON: Legacy	Stern	2011	5682
779	Taxi	Williams	1988	2505
686	Terminator 2: Judgment Day	Williams	1991	2524
687	The Addams Family	Bally	1992	20
2203	The Walking Dead (Pro)	Stern	2014	6155
2866	Total Nuclear Annihilation	Spooky	2017	6444
689	White Water	Williams	1993	2768
2277	Wrestlemania	Stern	2015	6215


```
>>> pinballmap --location 4495 --id-only loc_machines
1296,2832,2728,3022,738,2442,2571,2875,2924,695,2353,2306,1606,641,723,677,1276,2565,
→ 684,694,2844,1118,779,686,687,2203,2866,689,2277
```



```
>>> pinballmap machine_id 2571
```

id	name	manufacturer	year	ipdb_id
2571	Ghostbusters (LE)	Stern	2016	6334

1.4 Example Django settings.py

NOTE: Django settings, if present, will take precedence over arguments to `PinballMapClient(...)`

```
PINBALL_MAP = {
    'region_name': 'chicago', # a region name to use if not specified in code
    'location_id': your_location_id, # should be an int
    # email and token are required for all write operations
    'user_email': '...', # your pinball map account email, needed for write_
↪operations
    'user_password': '...', # your pinball map password, needed for write operations_
↪(not needed with token)
    'authentication_token': '...', # your pinball map api token, needed for write_
↪operations
    'cache_name': 'default', # default: 'default'
    'cache_key_prefix': 'pmap_', # default: 'pmap_'
}
```

1.5 Example Django management command

Create `yourapp/management/commands/update_pinball_map.py` and use this as a starting point:

```
from django.core.management.base import BaseCommand, CommandError
from pinballmap import PinballMapClient
from yourapp.somewhere import get_current_game_list

class Command(BaseCommand):
    help = 'Update the Pinball Map API. Adds/removes machines from our location.'

    def handle(self, *args, **options):
        try:
            games = get_current_game_list() # ← your code provides a list of current_
↪IDs
            # no args needed if you used Django settings as shown above:
            c = PinballMapClient()
            c.update_map([g.pinball_map_id for g in games])
            self.stdout.write(self.style.SUCCESS("Pinball Map updated."))
        except Exception as err:
            self.stderr.write(self.style.ERROR("Could not update pinball map because:
↪{}".format(err)))
```

2.1 0.3.4

- hopefully fix docs
- fix error from bumpversion

2.2 0.3.3

- requires Python 3.6
- CLI catches authentication errors more cleanly
- mostly code cleanups
- help outputs version # of python-pinballmap and Pinball Map API version supported
- all python code is now formatted using [black](#)

2.3 0.2.0

- breaking change: PinballMapClient now takes keyword arguments, old ordered argument syntax will no longer work
- now supports authentication tokens, signup process, getting auth details
- now uses https by default
- fix dry-run bug

2.4 0.1.2

- initial release

CHAPTER 3

Roadmap

- update command line interface to support signup and getting auth details
- eventually support all API actions, such as scores, machine conditions, etc.

PinballMapClient docs

class pinballmap.client.**PinballMapClient** (**kwargs)

Creates a PinballMapClient, optionally locked to a specific location_id and region name. It will use Django's default cache if installed and available.

Passed-in values are optional. Authentication Token and email is needed for all write operations.

You can use an instance to look up machines by name or id without specifying a location or region. Most other methods will fail without them.

If Django is installed and is in DEBUG mode, any write operations to the API will operate as a “dry run”. This is to prevent you from accidentally updating the map with inaccurate development data. (You're welcome.)

Note: the API uses names a bit inconsistently. . . sometimes it's user_token, sometimes it's authentication_token. Sometimes username, sometimes login. Always check the docs when adding/changing code here.

Parameters

- **authentication_token** – Your Pinball Map API Authentication Token, needed for all write operations.
- **user_email** – map account email
- **user_password** – map account password
- **location_id** – Your location_id, as found in the Pinball Map data
- **region_name** – Your region name, as found in the Pinball Map data
- **cache** – a cache object with get and set methods compatible with Django's cache
- **cache_name** – Django cache name to use. Default: 'default'
- **cache_key_prefix** – a prefix for cache keys. Default: 'pmap_'

add_machine (machine_id: int) → Optional[Dict]

Add a machine to my location.

NOTE: If it detects that Django is in debug mode, it will not actually perform the addition.

Parameters machine_id –

Returns JSON result or None.

auth_details (*username: str = None, password: str = None, update_self: bool = True*) → dict

Gets the authorization details for user. If login is successful, optionally set the token for this client (default is True).

If we have settings in Django, we'll use those automatically if username and password are None.

Parameters

- **username** – the username or email address
- **password** – the password
- **update_self** – whether to update this client instance's authentication_token, default is True

Returns result of auth_details request as dict

compare_location (*my_machine_ids: Iterable[int]*) → Dict

Compares a machine list with Pinball Map's data. Returns a dict with which ids to add, remove, or ignore (meaning they are already listed).

Parameters **my_machine_ids** – iterable of Pinball Map machine_ids

Returns { 'add': [id0, id1, idn...], 'remove': [id0, id1, idn...], 'ignore': [...] }

get_all_machines () → List[Dict]

Get list of all machines from PM DB. Cached to avoid a zillion large requests. Currently we are not storing results in the object because it's unlikely for an instance to be re-used for multiple name searches.

Returns list of every machine

get_location_machine_xrefs () → List[Dict]

Gets the list of location_machine_xrefs (LMXs) for our region, then filter it to include only ones from our location_id.

Since this API request is massive and slow, and likely to be accessed multiple times in the lifetime on an instance (when syncing, for example), we are using two levels of caching here. We store the filtered LMXs for our location in the PinballMapClient instance (in-memory), and ALSO cache the results for 15 minutes in Django's cache (if available). We are assuming no instance of this object will ever live long enough to be concerned about needing to bust its own copy of the cached data, but it will benefit from recent previous runs.

Returns list of LMXs for location_id

lmx_by_machine_id (*machine_id: int*) → Dict

Get the location_machine_xref for machine_id at my location.

Parameters **machine_id** –

Returns LMX, if found

machine_by_ipdb_id (*ipdb_id: int*) → Optional[Dict]

Find machine by IPDB number

Parameters **ipdb_id** – IPDB ID number

Returns pinball map data (as dict) or None if no match

machine_by_map_id (*map_id: int*) → Optional[Dict]

Find machine by pinball map ID

Parameters **map_id** – pinball map ID number

Returns pinball map data (as dict) or None if no match

machine_by_name (*query_string*: str, *min_score*: int = 2, *include_score*: bool = False) → Union[Tuple[Dict], Tuple[Tuple[Dict, str]]]

Finds likely name matches from the Pinball Map database and sorts results by a match quality score.

Parameters

- **query_string** – name of the game
- **min_score** – minimum quality score for matches. Our default of 2 seems to be the sweet spot.
- **include_score** – whether to include the match quality scores. Default = False.

Returns matches

machines_at_location (*location_id*: int = None) → List[Dict]

List the machines at location_id

Parameters **location_id** – optional location_id, or it will use the one in settings or set at init

Returns list of machines matching location_id

remove_machine (*machine_id*: int) → Optional[Dict]

Remove a machine from my location.

NOTE: If it detects that Django is in debug mode, it will not actually perform the removal.

Parameters **machine_id** – machine id to remove

Returns JSON result of operation (as a dict) or None

signup_user (*username*: str, *email*: str, *password*: str, *update_self*: bool = True) → dict

Creates a user account. Note: This is an easy way to get a token to use later. If login is successful, optionally set the token for this client.

If signup fails, attempt to log in and self-update with the same credentials in case the reason for failure is simply that the account already exists.

NOTE: the API server will send an email with a verification link. You can't do anything further until confirmed.

Example responses:

Parameters

- **username** – the username
- **email** – user's email address
- **password** – the password
- **update_self** – whether to update this client instance's authentication_token, default is True

Returns result of auth_details request as dict

update_map (*machine_ids*: Iterable[int]) → Dict

Given a complete list of machine_ids for the location, this will add and remove them as needed so that Pinball Map matches your current list of machines.

Parameters **machine_ids** – the pinball map id numbers for your current list of machines

Returns dict of count of machines added, removed, or ignored

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

p

`pinballmap.client`, 9

A

`add_machine()` (pinballmap.client.PinballMapClient method), [9](#)

`auth_details()` (pinballmap.client.PinballMapClient method), [10](#)

C

`compare_location()` (pinballmap.client.PinballMapClient method), [10](#)

G

`get_all_machines()` (pinballmap.client.PinballMapClient method), [10](#)

`get_location_machine_xrefs()` (pinballmap.client.PinballMapClient method), [10](#)

L

`lmx_by_machine_id()` (pinballmap.client.PinballMapClient method), [10](#)

M

`machine_by_ipdb_id()` (pinballmap.client.PinballMapClient method), [10](#)

`machine_by_map_id()` (pinballmap.client.PinballMapClient method), [10](#)

`machine_by_name()` (pinballmap.client.PinballMapClient method), [10](#)

`machines_at_location()` (pinballmap.client.PinballMapClient method), [11](#)

P

`pinballmap.client` (module), [9](#)

`PinballMapClient` (class in pinballmap.client), [9](#)

R

`remove_machine()` (pinballmap.client.PinballMapClient method), [11](#)

S

`signup_user()` (pinballmap.client.PinballMapClient method), [11](#)

U

`update_map()` (pinballmap.client.PinballMapClient method), [11](#)