dota2api Documentation

Release 1

Joshua Duffy

Feb 28, 2019

Contents

Welcome to the dota2api documentation. This Python library is an unofficial wrapper for the Dota 2 API[1] from Valve Software[2]. The repository can be found on GitHub[3].

[1] https://wiki.teamfortress.com/wiki/WebAPI#Dota_2
[2] http://www.valvesoftware.com/
[3] https://github.com/joshuaduffy/dota2api

CHAPTER 1

Contents

1.1 Installation

This section covers installation of the library.

Tip: Work in a virtual environment!

1.1.1 Pip

Installing via pip[1] is the recommended method:

```
$ pip install dota2api
```

1.1.2 Build from source

You can also download the latest version of the code from the repository[2] and install:

```
$ git clone https://github.com/joshuaduffy/dota2api/ && cd dota2api/
$ python setup.py install
```

1.2 Tutorial

This section covers basic usage of the library.

[1] http://www.pip-installer.org/
[2] https://github.com/joshuaduffy/dota2api

1.2.1 Getting an API Key

Get one from Valve.

1.2.2 D2_API_KEY environment variable

You can set the `D2_API_KEY` environment variable to save entering it all the time.

For example, in Linux:

```
$  export D2_API_KEY=83247983248793298732
```

1.2.3 Initialising

If you've set the API Key as an environment variable, initialise the module like so:

```
>>> import dota2api
>>> api = dota2api.Initialise()
```

If not you'll need to pass it into the constructor:

```
>>> import dota2api
>>> api = dota2api.Initialise("45735474375437457457")
```

Official DOTA2 web API would response identifiers for records like heroes, items, lobby type, game mode, etc. By default, this dota2api would translate most dota2 identifiers into human readable strings. But you can disable our translation by enabling raw mode:

```
>>> import dota2api
>>> api = dota2api.Initialise("45735474375437457457", raw_mode=True)
```

By default, you'll get {"hero_name": "axe"} for axe but when raw_mode is on, it will be replaced by {"hero_id", 2}.

1.2.4 API calls

The functions are mapped to API calls:

```
>>> match = api.get_match_details(match_id=1000193456)
```

The responses are then returned in a `dict`:

```
>>> match['radiant_win']
False
```

Parameters can be used to filter the results. They're all listed in the *Library Reference*

1.2.5 Get match history

You can use the `account_id` parameter to filter the results for a specific user.

```
>>> hist = api.get_match_history(account_id=76482434)
```

1.2.6 Get match details

```
>>> match = api.get_match_details(match_id=1000193456)
```

1.2.7 Other API calls

Listed in the *Library Reference*

1.2.8 Exceptions

`APIError` will be raised if an error message is returned by the API.

`APITimeoutError` will be raised you're making too many requests or the API itself is down.

`APIAuthenticationError` will be raised if you're using an invalid API key.

1.3 Responses

This section describes the dictionary structure of each response.

Every response has a number of attributes you can use. For example:

```
>>> match = api.get_match_details(match_id=1000193456)
```

The following will return the URL constructed by the library:

```
>>> match.url
```

The following will return the response as raw json:

```
>>> match.json
```

1.3.1 get_match_history()

Returns a dictionary with a list of `players` within.

`match::lobby_type` – see *lobby_type*.

`player::player_slot` – see *player_slot*.

```
{
    num_results                 - Number of matches within a single response
    total_results               - Total number of matches for this query
    results_remaining           - Number of matches remaining to be retrieved with␣
↪subsequent API calls
    [matches]                   - List of matches for this response
    {
        match_id                - Unique match ID
        match_seq_num           - Number indicating position in which this match was␣
↪recorded
        start_time              - Unix timestamp of beginning of match
        lobby_type              - See lobby_type table
```

(continues on next page)

(continued from previous page)

```
        [player]            - List of players in the match
        {
            account_id      - Unique account ID
            player_slot     - Player's position within the team
            hero_id         - Unique hero ID
        }
    }
}
```

player_slot

The player slot is an 8-bit representation of the player's team and the slot (0-4) within the team.

```
┌─────────────── Team (false if Radiant, true if Dire).
│ ┌─────────── Not used.
│ │ │ │ │ ┌─── The position of a player within their team (0-4).
│ │ │ │ │ │ │ │
0 0 0 0 0 0 0 0
```

1.3.2 get_match_history_by_seq_num()

Returns a dictionary with a list of `matches` within. See *get_match_details()* for structure of matches.

```
{
    status
        1 - Success
        8 - Matches_requested must be greater than 0
    statusDetail        - Message explaining a status that is not equal to 1
    [matches]           - See get_match_details()
}
```

1.3.3 get_match_details()

Returns a `match` dictionary with `players`.

For dynamic values such as kills or gold, if the match is live, then the value is current as of the API call. For matches that have finished, these values are simply the value at the end of the match for the player.

`lobby_type` – see *lobby_type*.

`game_mode` and `game_mode_name` – see *game_mode*

```
{
    season                  - Season the game was played in
    radiant_win             - Win status of game (True for Radiant win, False for␣
↪Dire win)
    duration                - Elapsed match time in seconds
    start_time              - Unix timestamp for beginning of match
    match_id                - Unique match ID
    match_seq_num           - Number indicating position in which this match was␣
↪recorded
    tower_status_radiant    - Status of Radiant towers
    tower_status_dire       - Status of Dire towers
```

(continues on next page)

(continued from previous page)

```
    barracks_status_radiant - Status of Radiant barracks
    barracks_status_dire    - Status of Dire barracks
    cluster                 - The server cluster the match was played on, used in␣
↪retrieving replays
    cluster_name            - The region the match was played on
    first_blood_time        - Time elapsed in seconds since first blood of the match
    lobby_type              - See lobby_type table
    lobby_name              - See lobby_type table
    human_players           - Number of human players in the match
    leagueid                - Unique league ID
    positive_votes          - Number of positive/thumbs up votes
    negative_votes          - Number of negative/thumbs down votes
    game_mode               - See game_mode table
    game_mode_name          - See game_mode table
    radiant_captain         - Account ID for Radiant captain
    dire_captain            - Account ID for Dire captain
    [pick_bans]
    {
        {
            hero_id         - Unique hero ID
            is_pick         - True if hero was picked, False if hero was banned
            order           - Order of pick or ban in overall pick/ban sequence
            team            - See team_id table.

        }
    }
    [players]
    {
        account_id          - Unique account ID
        player_slot         - Player's position within the team
        hero_id             - Unique hero ID
        hero_name           - Hero's name
        item_#              - Item ID for item in slot # (0-5)
        item_#_name         - Item name for item in slot # (0-5)
        kills               - Number of kills by player
        deaths              - Number of player deaths
        assists             - Number of player assists
        leaver_status       - Connection/leaving status of player
        gold                - Gold held by player
        last_hits           - Number of last hits by player (creep score)
        denies              - Number of denies
        gold_per_min        - Average gold per minute
        xp_per_min          - Average XP per minute
        gold_spent          - Total amount of gold spent
        hero_damage         - Amount of hero damage dealt by player
        tower_damage        - Amount of tower damage dealt by player
        hero_healing        - Amount of healing done by player
        level               - Level of player's hero
        [ability_upgrades]  - Order of abilities chosen by player
        {
            ability         - Ability chosen
            time            - Time in seconds since match start when ability was␣
↪upgraded
            level           - Level of player at time of upgrading
        }

        [additional_units]  - Only available if the player has a additional unit
```

(continues on next page)

(continued from previous page)

```
        {
            unitname        - Name of unit
            item_#          - ID of item in slot # (0-5)
        }
    }
    // These fields are only available for matches with teams //
    [radiant_team]
    {
        team_name             - Team name for Radiant
        team_logo             - Team logo for Radiant
        team_complete         - ?
    }
    [dire_team]
    {
        team_name             - Team name for Dire
        team_logo             - Team logo for Dire
        team_team_complete    - ?
    }
}
```

1.3.4 get_league_listing()

Returns a dictionary with a list of `leagues` within; can be viewed with DotaTV.

```
{
    [leagues]
    {
        description     - Description of the league
        itemdef         - ID for an item associated with the tournament
        leagueid        - Unique league ID
        name            - Name of the league
        tournament_url  - League website information
    }
}
```

1.3.5 get_live_league_games()

Returns a dictionary with a list of league `games` within.

`league_tier` – see league_tier.

`tower_state` – see *Towers and Barracks*.

`series_type` – see *series_type*.

`player::team` – see *team_id*.

```
{
    [games]
    {
        league_id                 - ID for the league in which the match is being played
        league_tier               - Level of play in this game
        league_series_id          - ?
        [players]                 - list of all players in the match
        {
```

(continues on next page)

(continued from previous page)

```
            account_id           - Unique account ID
            name                 - In-game display name
            hero_id              - Unique hero ID
            team                 - Team the player is on
        }
        series_id                - ID for the game series
        series_type              - Type of tournament series
        stage_name               - ?
        game_number              - Game number of the series
        radiant_series_wins      - Number of wins by Radiant during the series
        dire_series_wins         - Number of wins by Dire during the series
        tower_state              - State of *all* towers in the match
        spectators               - Number of spectators watching
        lobby_id                 - ID for the match's lobby
        stream_delay_s           - Delay in seconds for streaming to spectators

        // These fields are only available for matches with teams //
        [radiant_team]
        {
            team_name              - Team name for Radiant
            team_logo              - Team logo for Radiant
            team_complete          - ?
        }
        [dire_team]
        {
            team_name              - Team name for Dire
            team_logo              - Team logo for Dire
            team_team_complete     - ?
        }
    }
}
```

1.3.6 get_team_info_by_team_id()

Returns a dictionary with a list of `teams` within.

```
{
    status                                  - 1 if success, non-1 otherwise
    [teams]
    {
        admin_account_id                    - Account ID for team admin
        calibration_games_remaining         - ?
        country_code                        - ISO 3166-1 country code
        games_played                        - Number of games played by team with␣
↪current team members
        league_id_#                         - League IDs in which the team has played
        logo                                - UGC ID for the team logo
        logo_sponsor                        - UGC ID for the team sponsor logo
        name                                - Team's name
        player_#_account_id                 - Account ID for player # (0-5)
        tag                                 - Team's tag
        team_id                             - Unique team ID
        time_created                        - Unix timestamp of team creation
        url                                 - Team-provided URL
    }
```

(continues on next page)

(continued from previous page)

```
}
```

1.3.7 get_player_summaries()

Returns a dictionary with a list of `players` within.

```
{
    [player]
    {
        avatar                      - 32x32 avatar image
        avatarfull                  - 184x184 avatar image
        avatarmedium                - 64x64 avatar image
        communityvisibilitystate    - See table below.
        lastlogoff                  - Unix timestamp since last time logged out of␣
↪steam
        personaname                 - Equivalent of Steam username
        personastate                - See table below.
        personastateflags           - ?
        primaryclanid               - 64-bit unique clan identifier
        profilestate                - ?
        profileurl                  - Steam profile URL
        realname                    - User's given name
        steamid                     - Unique Steam ID
        timecreated                 - Unix timestamp of profile creation time
    }
}
```

communityvisibilitystate

Value	Description
1	Private
2	Friends only
3	Friends of friends
4	Users only
5	Public

personastate

Value	Description
0	Offline
1	Online
2	Busy
3	Away
4	Snooze
5	Looking to trade
6	Looking to play

1.3.8 get_heroes()

```
{
    count                           - Number of results
    status                          - HTTP status code
    [heroes]
    {
        id                          - Unique hero ID
        name                        - Hero's name
        localized_name              - Localized version of hero's name
        url_full_portrait           - URL to full-size hero portrait (256x144)
        url_large_portrait          - URL to large hero portrait (205x115)
        url_small_portrait          - URL to small hero portrait (59x33)
        url_vertical_portrait       - URL to vertical hero portrait (235x272)
    }
}
```

1.3.9 get_game_items()

```
{
    count                   - Number of results
    status                  - HTTP status respose
    [items]
    {
        id                  - Unique item ID
        name                - Item's name
        cost                - Item's gold cost in game, 0 if recipe
        localized_name      - Item's localized name
        recipe              - True if item is a recipe item, false otherwise
        secret_shop         - True if item is bought at the secret shop, false otherwise
        side_shop           - True if item is bought at the side shop, false otherwise
    }
}
```

1.3.10 get_tournament_prize_pool()

```
{
    league_id    - Unique league ID
    prizepool    - Current prize pool if the league includes a community-funded pool,␣
↪otherwise 0
    status       - HTTP status code
}
```

1.3.11 Towers and Barracks

Combined status

The overall match tower and barracks status uses 32 bits for representation and should be interpreted as follows:

```
┌─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─────────────────────────────────────────── Not used.
│ │ │ │ │ │ │ │ │ │ └─────────────────────────────────────────── Dire Ancient Top
```

(continues on next page)

(continued from previous page)

```
│ │ │ │ │ │ │ │ │ │ │ ───────────────────────────────────────── Dire Ancient Bottom
│ │ │ │ │ │ │ │ │ │ │ │ ─────────────────────────────────────── Dire Bottom Tier 3
│ │ │ │ │ │ │ │ │ │ │ │ │ ───────────────────────────────────── Dire Bottom Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────────────────────────────────── Dire Bottom Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───────────────────────────────── Dire Middle Tier 3
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────────────────────────────── Dire Middle Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───────────────────────────── Dire Middle Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────────────────────────── Dire Top Tier 3
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───────────────────────── Dire Top Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────────────────────── Dire Top Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───────────────────── Radiant Ancient Top
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────────────────── Radiant Ancient Bottom
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───────────────── Radiant Bottom Tier 3
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────────────── Radiant Bottom Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───────────── Radiant Bottom Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────────── Radiant Middle Tier 3
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───────── Radiant Middle Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─────── Radiant Middle Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ───── Radiant Top Tier 3
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─── Radiant Top Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─ Radiant Top Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Single team tower status

The tower status for a single team uses 16 bits for representation and should be interpreted as follows:

```
───────────────────────────── Not used.
│ │ │ │ │ ───────────────────── Ancient Bottom
│ │ │ │ │ │ ─────────────────── Ancient Top
│ │ │ │ │ │ │ ───────────────── Bottom Tier 3
│ │ │ │ │ │ │ │ ─────────────── Bottom Tier 2
│ │ │ │ │ │ │ │ │ ───────────── Bottom Tier 1
│ │ │ │ │ │ │ │ │ │ ─────────── Middle Tier 3
│ │ │ │ │ │ │ │ │ │ │ ───────── Middle Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ ─────── Middle Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ ───── Top Tier 3
│ │ │ │ │ │ │ │ │ │ │ │ │ │ ─── Top Tier 2
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ ─ Top Tier 1
│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Single team barracks status

The barracks status uses 8 bits for representation and should be interpreted as follows:

```
────────────── Not used.
│ │ ─────────── Bottom Ranged
│ │ │ ───────── Bottom Melee
│ │ │ │ ─────── Middle Ranged
│ │ │ │ │ ───── Middle Melee
│ │ │ │ │ │ ─── Top Ranged
│ │ │ │ │ │ │ ─ Top Melee
```

(continues on next page)

(continued from previous page)

```
| | | | | | | |
0 0 0 0 0 0 0 0
```

1.3.12 Status code mappings

These tables outline various codes/status in responses and their meaning.

See `dota2api.parse` for various parsing utilities.

series_type

Value	Description
0	Non-series
1	Best of 3
2	Best of 5

league_tier

Value	Description
1	Amateur
2	Professional
3	Premier

game_mode

Value	Description
0	Unknown
1	All Pick
2	Captain’s Mode
3	Random Draft
4	Single Draft
5	All Random
6	Intro
7	Diretide
8	Reverse Captain’s Mode
9	The Greeviling
10	Tutorial
11	Mid Only
12	Least Played
13	New Player Pool
14	Compendium Matchmaking
15	Custom
16	Captains Draft
17	Balanced Draft
18	Ability Draft
19	Event (?)
20	All Random Death Match
21	Solo Mid 1 vs 1
22	Ranked All Pick

lobby_type

Status	Description
-1	Invalid
0	Public matchmaking
1	Practice
2	Tournament
3	Tutorial
4	Co-op with AI
5	Team match
6	Solo queue
7	Ranked matchmaking
8	Solo Mid 1 vs 1

leaver_status

ID	Value	Description
0	NONE	finished match, no abandon
1	DISCONNECTED	player DC, no abandon
2	DISCONNCECTED_TOO_LONG	player DC > 5min, abandon
3	ABANDONED	player dc, clicked leave, abandon
4	AFK	player AFK, abandon
5	NEVER_CONNECTED	never connected, no abandon
6	NEVER_CONNECTED_TOO_LONG	too long to connect, no abandon

team_id

Value	Description
0	Radiant
1	Dire
2	Broadcaster
3+	Unassigned (?)

1.3.13 get_top_live_games()

Returns a dictionary that includes top MMR live games

```
{
    [game_list]
    {
        activate_time           -
        deactivate_time         -
        server_steam_id         -
        lobby_id                -
        league_id               - League ID (Available for league matches)
        lobby_type              - See lobby_type table
        game_time               - Current in-game time (in seconds)
        delay                   - Delay in seconds for spectators
        spectators              - Number of spectators in-game
        game_mode               - See game_mode table
        average_mmr             - Average MMR of players in the game
        team_name_radiant       - Radiant team name (Available for matches with teams)
        team_name_dire          - Dire team name (Available for matches with teams)
        sort_score              -
        last_update_time        -
        radiant_lead            - Gold lead for Radiant (negative if Dire leads)
        radiant_score           - Radiant kill score
        dire_score              - Dire kill score
        building_state          -
        [players]
        {
            account_id          - Player's 32-bit Steam ID
            hero_id             - Player's hero ID
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

1.4 Library Reference

This section covers the `dota2api` package, the `parse` module and the `exceptions` used.

1.4.1 API

1.4.2 Parser

1.4.3 Exceptions

1.5 Contributing

This section provides help for people who wish to contribute to the project.

We are open to most change requests, the only request is that every piece of functionality is accompanied by a test!

1.5.1 Documentation

Documentation improvements are always welcome, I'm hoping this will be a useful guide to the API as most information online is out of date.

1.5.2 Bug reports

Forks or bug reports are welcome! If you spot any errors in the code or documentation please open an issue on GitHub.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

CHAPTER 3

References