d2api Documentation

Release 1.1

Raghav Sairam

Jul 25, 2019

Contents

This Python library is an unofficial wrapper and parser for the Dota 2 Web API by Valve.

CHAPTER 1

Installation

1.1 Install using pip (recommended)

Install d2api from pip using:

```
$ pip install d2api
```

1.2 Build from source

This would install the latest version of d2api. You can download the latest version of the code from the git repository and run:

```
$ git clone https://github.com/whoophee/d2api/ && cd d2api/
$ python setup.py install
```

CHAPTER 2

Tutorial

2.1 Getting started

2.1.1 Getting an API key

First and foremost, you'll need a Steam API key. You can get one here.

2.1.2 Initialize wrapper via environment variable

Create a new environment variable `D2_API_KEY` and set its value to the API key. You can then, just use the following code to initialize the wrapper.

```
api = d2api.APIWrapper()
```

2.1.3 Initialize wrapper inline

Literally just initialize the wrapper inline. That's about it.

```
# overrides the environment variable key
api = d2api.APIWrapper('YOUR_API_KEY')
```

2.1.4 Unparsed response

There's a good chance you'd like your responses au naturel. Just set `parse_response = False`. The wrapper returns the response text as is (without using the built-in json parser).

```
api = d2api.APIWrapper(api_key = 'YOUR_API_KEY', parse_response = False)
```

Note: While it is highly recommended that a json response have unique key-value pairs, it is not mandatory that they be unique. Some responses of the Steam WebAPI consists of such repeated key-value pairs. Use `d2api.src.util.decode_json` to parse these results to avoid losing content.

2.2 Examples

2.2.1 Hero frequency in last 100 games

```
import d2api
from d2api.src import entities

api = d2api.APIWrapper()

# fetch latest matches
match_history = api.get_match_history()

# get frequency of heroes played in the latest 100 games
heroes = {}

for match in match_history['matches']:
    for player in match['players']:
        hero_id = player['hero']['hero_id']
        if not hero_id in heroes:
            heroes[hero_id] = 0
        heroes[hero_id] += 1

# print hero frequency by name
for hero_id, freq in heroes.items():
    print(entities.Hero(hero_id)['hero_name'], freq)
```

2.2.2 Using the API without the API

```
from d2api.src import entities

# Hero/Item/Ability information is available without having to specify a key
print(entities.Hero(67)['hero_name'])
print(entities.Item(208)['item_aliases'])
print(entities.Ability(6697)['ability_name'])

# Use steam32/steam64 IDs interchangeably
steam_account = entities.SteamAccount(1020002)
print(steam_account['id32'], steam_account['id64'])
```

2.2.3 Matches without leavers

```
# Fetch last 100 very high skill games and filter out games that have leavers
import d2api
```

(continues on next page)

(continued from previous page)

```
api = d2api.APIWrapper()
vhs = api.get_match_history(skill = 3)

matches = [api.get_match_details(m['match_id']) for m in vhs['matches']]

# now filter out matches that have leavers
matches = [m for m in matches if not m.has_leavers()]

# number of matches that remain
print(len(matches))

# print the first match
print(matches[0])
```

CHAPTER 3

Quick Reference

Below, is the generic dictionary access structure for any of the given endpoints.

3.1 get_match_history()

```
{
    list(matches): {
        dire_team_id,
        lobby_type,
        match_id,
        match_seq_num,
        list(players): {
            hero: {hero_id, hero_name},
            side,
            steam_account: {id32, id64}
        },
        radiant_team_id,
        start_time
    },
    num_results,
    results_remaining,
    status,
    total_results
}
```

3.2 get_match_details()

```
{
    cluster,
    dire_buildings: {
```

(continues on next page)

(continued from previous page)

```
        barracks_status,
        bot_ancient,
        bot_melee,
        bot_ranged,
        bot_t1,
        bot_t2,
        bot_t3,
        mid_melee,
        mid_ranged,
        mid_t1,
        mid_t2,
        mid_t3,
        top_ancient,
        top_melee,
        top_ranged,
        top_t1,
        top_t2,
        top_t3,
        tower_status
    },
    dire_score,
    duration,
    engine,
    first_blood_time,
    flags,
    game_mode,
    human_players,
    leagueid,
    lobby_type,
    match_id,
    match_seq_num,
    negative_votes,
    list(picks_bans): {
        hero: {hero_id, hero_name},
        is_pick,
        order,
        side
    },
    list(players): {
        ability_upgrades,
        list(additional_units): {
            list(backpack): {
                item_aliases,
                item_cost,
                item_id,
                item_name
            },
            list(inventory): {
                list(item_aliases),
                item_cost,
                item_id,
                item_name
            },
            unitname
        },
        assists,
        list(backpack): {
```

(continues on next page)

(continued from previous page)

```
            list(item_aliases),
            item_cost,
            item_id,
            item_name
        },
        deaths,
        denies,
        gold_per_min,
        hero: {hero_id, hero_name},
        list(inventory): {
            list(item_aliases),
            item_cost,
            item_id,
            item_name
        },
        kills,
        last_hits,
        leaver_status,
        level,
        side,
        steam_account: {id32, id64},
        xp_per_min
    },
    list(players_minimal): {
        hero: {hero_id, hero_name},
        side,
        steam_account: {id32, id64}
    },
    positive_votes,
    pre_game_duration,
    radiant_buildings: {
        barracks_status,
        bot_ancient,
        bot_melee,
        bot_ranged,
        bot_t1,
        bot_t2,
        bot_t3,
        mid_melee,
        mid_ranged,
        mid_t1,
        mid_t2,
        mid_t3,
        top_ancient,
        top_melee,
        top_ranged,
        top_t1,
        top_t2,
        top_t3,
        tower_status
    },
    radiant_score,
    start_time,
    winner
}
```

3.3 get_heroes()

```
{
    count,
    list(heroes): {
        id,
        localized_name,
        name
    },
    status
}
```

3.4 get_game_items()

```
{
    list(game_items): {
        cost,
        id,
        localized_name,
        name,
        recipe,
        secret_shop,
        side_shop
    },
    status
}
```

3.5 get_tournament_prize_pool()

```
{
    league_id,
    prize_pool,
    status
}
```

3.6 get_top_live_game()

```
{
    list(game_list): {
        activate_time,
        average_mmr,
        deactivate_time,
        delay,
        dire_score,
        dire_team: {dire_id, dire_name},
        dire_towers: {
            bot_ancient,
            bot_t1,
```

(continues on next page)

(continued from previous page)

```
            bot_t2,
            bot_t3,
            mid_t1,
            mid_t2,
            mid_t3,
            top_ancient,
            top_t1,
            top_t2,
            top_t3,
            tower_status
        },
        game_mode,
        game_time,
        last_update_time,
        league_id,
        lobby_id,
        lobby_type,
        match_id,
        list(players): {
            hero: {hero_id, hero_name},
            steam_account: {id32, id64}
        },
        radiant_lead,
        radiant_score,
        radiant_team: {team_id, team_name},
        radiant_towers: {
            bot_ancient,
            bot_t1,
            bot_t2,
            bot_t3,
            mid_t1,
            mid_t2,
            mid_t3,
            top_ancient,
            top_t1,
            top_t2,
            top_t3,
            tower_status
        },
        series_id,
        server_steam_id,
        sort_score,
        spectators,
        team_logo_dire,
        team_logo_radiant
    }
}
```

3.7 get_team_info_by_team_id()

```
{
    status,
    list(teams): {
        admin_account_id,
```

(continues on next page)

(continued from previous page)

```
        calibration_games_remaining,
        country_code,
        games_played,
        logo,
        logo_sponsor,
        name,
        player_0_account_id,
        player_1_account_id,
        player_2_account_id,
        player_3_account_id,
        player_4_account_id,
        player_5_account_id,
        tag,
        time_created,
        url
    }
}
```

3.8 get_live_league_games()

```
{
    list(games): {
        dire_series_wins,
        dire_team: {
            complete,
            team_id,
            team_logo,
            team_name
        },
        league_id,
        league_node_id,
        lobby_id,
        match_id,
        list(players): {
            hero: {hero_id, hero_name},
            name,
            side,
            steam_account: {id32, id64}
        },
        radiant_series_wins,
        radiant_team: {
            complete,
            team_id,
            team_logo,
            team_name
        },
        scoreboard: {
            dire: {
                list(bans): {hero_id, hero_name},
                barracks_state,
                buildings: {
                    barracks_status,
                    bot_ancient,
                    bot_melee,
```

(continues on next page)

(continued from previous page)

```
                bot_ranged,
                bot_t1,
                bot_t2,
                bot_t3,
                mid_melee,
                mid_ranged,
                mid_t1,
                mid_t2,
                mid_t3,
                top_ancient,
                top_melee,
                top_ranged,
                top_t1,
                top_t2,
                top_t3,
                tower_status
            },
            list(picks): {hero_id, hero_name},
            list(players): {
                list(abilities): {
                    ability: {ability_id, ability_name},
                    ability_level
                },
                assists,
                deaths,
                denies,
                gold,
                gold_per_min,
                hero: {hero_id, hero_name},
                list(inventory): {
                    list(item_aliases),
                    item_cost,
                    item_id,
                    item_name
                },
                kills,
                last_hits,
                level,
                net_worth,
                player_slot,
                position_x,
                position_y,
                respawn_timer,
                steam_account: {id32, id64},
                ultimate_cooldown,
                ultimate_state,
                xp_per_min
            },
            score,
            tower_state
        },
        duration,
        radiant: {
            list(bans): {hero_id, hero_name},
            barracks_state,
            buildings: {
                barracks_status,
```

(continues on next page)

(continued from previous page)

```
                    bot_ancient,
                    bot_melee,
                    bot_ranged,
                    bot_t1,
                    bot_t2,
                    bot_t3,
                    mid_melee,
                    mid_ranged,
                    mid_t1,
                    mid_t2,
                    mid_t3,
                    top_ancient,
                    top_melee,
                    top_ranged,
                    top_t1,
                    top_t2,
                    top_t3,
                    tower_status
                },
                list(picks): {hero_id, hero_name},
                list(players): {
                    list(abilities): {
                        ability: {ability_id, ability_name},
                        ability_level
                    },
                    assists,
                    deaths,
                    denies,
                    gold,
                    gold_per_min,
                    hero: {hero_id, hero_name},
                    list(inventory): {
                        list(item_aliases),
                        item_cost,
                        item_id,
                        item_name
                    },
                    kills,
                    last_hits,
                    level,
                    net_worth,
                    player_slot,
                    position_x,
                    position_y,
                    respawn_timer,
                    steam_account: {id32, id64},
                    ultimate_cooldown,
                    ultimate_state,
                    xp_per_min
                },
                score,
                tower_state
            },
            roshan_respawn_timer
        },
        series_type,
        spectators,
```

(continues on next page)

(continued from previous page)

```
        stream_delay_s
    },
    status
}
```

3.9 get_broadcaster_info()

```
{
    live,
    server_steam_id,
    steam_account: {id32, id64}
}
```

3.10 get_player_summaries()

```
{
    list(players): {
        avatar,
        avatarfull,
        avatarmedium,
        communityvisibility,
        lastlogoff,
        personaname,
        personastate,
        personastateflags,
        primaryclanid,
        profilestate,
        profileurl,
        realname,
        steam_account: {id32, id64},
        steamid,
        timecreated
    }
}
```

CHAPTER 4

Endpoints

class d2api.**APIWrapper**(*api_key=None*, *parse_response=True*, *requests_per_second=1*)

Wrapper initialization requires either environment variable `D2_API_KEY` be set, or `api_key` be provided as an argument.

Parameters

- **api_key** (*str*) – Steam API key
- **parse_response** (*bool*) – set to `False` to get an unparsed json string
- **requests_per_second** (*int*) – rate limit requests to send requests politely (set to `-1` to ignore rate limiting)

get_broadcaster_info(***kwargs*)

Get the broadcasting status of a user.

Parameters

- **account_id** (*int*) – 32/64-bit account ID
- **steam_account** (`SteamAccount`) – Used in place of account_id

Returns Broadcasting information of a user.

Return type *BroadcasterInfo*

get_game_items(***kwargs*)

Get a list of items in Dota 2.

Parameters **language** (*string, optional*) – The language to provide hero names in

Returns Item information.

Return type *GameItems*

get_heroes(***kwargs*)

Get a list of heroes in Dota 2.

Parameters

- **language** (*string, optional*) – The language to provide hero names in

- **itemizedonly** (*bool, optional*) – Return a list of itemized heroes only

Returns Hero information.

Return type *Heroes*

get_live_league_games (***kwargs*)
Get a list of in-progress league matches, as well as their details at the time of query.

Returns Details of in-progress live league games.

Return type *LiveLeagueGames*

get_match_details (*match_id, **kwargs*)
Get detailed information about a particular match.

Parameters match_id (*int, string*) – Match ID

Returns Details of a match.

Return type *MatchDetails*

get_match_history (***kwargs*)
Get a list of matches, filtered by various parameters.

Parameters

- **hero_id** (*int, optional*) – Matches containing this hero. A list of hero IDs can be fetched via the `get_heroes()` method
- **hero** (`Hero`, *optional*) – Used in place of hero_id
- **game_mode** (*int, optional*) – Games of this game mode are fetched
- **skill** (*int, optional*) – Skill bracket for the matches (Ignored if an account ID is specified)
- **min_players** (*int, optional*) – Minimum amount of players in a match for the match to be returned.
- **account_id** (*int, optional*) – 32/64-bit account ID
- **steam_account** (`SteamAccount`, *optional*) – Used in place of account_id
- **league_id** (*int, optional*) – Only return matches from this league. get_league_listing() has been discontinued
- **start_at_match_id** (*int, optional*) – Start searching for matches equal to or older than this match ID
- **matches_requested** (*int, optional*) – Defaults to *100*
- **tournament_games_only** (*int, optional*) – 0 = False, 1 = True

Returns Information of matches.

Return type *MatchHistory*

get_match_history_by_sequence_num (***kwargs*)
Get a list of matches ordered by sequence number. Uses a parser similar to that of `get_match_history()` method

Parameters

- **start_at_match_seq_num** (*int*) – The match sequence number to start returning results from
- **matches_requested** (*int, optional*) – Defaults to *100*

Returns Information of matches.

Return type *MatchHistory*

get_player_summaries(***kwargs*)
Get Steam details of users.

Parameters

- **account_ids** (*list (int)*) – 32/64-bit account ID
- **steam_accounts** (*list (SteamAccount)*) – Used in place of account IDs

Returns Information of steam accounts

Return type *PlayerSummaries*

get_team_info_by_team_id(***kwargs*)
Get a list of teams' information.

Parameters

- **start_at_team_id** (*int, optional*) – The team id to start returning results from
- **teams_requested** (*int, optional*) – The amount of teams to return

Returns A list of teams' information.

Return type *TeamInfoByTeamID*

get_top_live_game(*partner=0*, ***kwargs*)
Get details of on-going live games.

Parameters **partner** (*int, optional*) – Which partner's games to use (default *0*)

Returns Details of on-going live games.

Return type *TopLiveGame*

get_tournament_prize_pool(***kwargs*)
Get the current prizepool of specific tournaments.

Parameters **leagueid** (*int*) – The ID of the league to get the prize pool of

Returns Prizepool of a tournament.

Return type *TournamentPrizePool*

d2api.**update_local_data**(*purge=True*)
Synchronize local data with current repository data

Parameters **purge** (*bool*) – Set to `True` to delete local content

CHAPTER 5

Documentation

There are several response types/entities accross the different endpoints of the Dota 2 WebAPI. This API aims at removing inconsistencies and unifying the response content.

All variables belonging to a class are accessed using the `__getitem__` method (similar to a dict). See *examples* for more details.

5.1 get_match_history()

class `d2api.src.wrappers.`**`MatchHistory`**(*response_text*)
`get_match_history` or *`get_match_history_by_sequence_num`* response object

Variables **`matches`** (*`list`* (`MatchSummary`)) – List of match summaries

class `d2api.src.wrappers.`**`MatchSummary`**(*default_obj*)
A brief summary of queried games

Variables

- **`match_id`** (*`int`*) – The unique ID of a match
- **`match_seq_num`** (*`int`*) – Represents the sequence in which matches were recorded
- **`start_time`** (*`int`*) – Unix timestamp of game begin time
- **`lobby_type`** (*`int`*) – Integer representing type of lobby
- **`players`** (*`list`* (`PlayerMinimal`)) – List of player summaries

5.2 get_match_details()

class `d2api.src.wrappers.`**`MatchDetails`**(*response_text*)
`get_match_details` response object

Variables

- **players** (PlayerUnit) – List of players in the game
- **players_minimal** (PlayerMinimal) – List of players represented minimally
- **picks_bans** (PickBan) – List of picks/bans
- **season** (*int*) – The season in which the game was played
- **winner** (*str*) – Side that won the game (radiant/dire)
- **duration** (*int*) – Duration of the game (in seconds)
- **pre_game_duration** (*int*) – Duration for game to begin (in seconds)
- **start_time** (*int*) – Unix timestamp of match start
- **match_seq_num** (*int*) – Number denoting the order in which matches were recorded
- **radiant_buildings** (Buildings) – Radiant building statuses at the end of the game
- **dire_buildings** (Buildings) – Dire building statuses at the end of the game
- **cluster** (*int*) – The server cluster the match was played upon (used to fetch replays)
- **first_blood_time** (*int*) – Time of first-blood occurrance
- **lobby_type** (*int*) – Type of lobby
- **human_players** (*int*) – Number of human players in the game
- **leagueid** (*int*) – The league that this match was a part of
- **positive_votes** (*int*) – The number of thumbs-up the game has received by users
- **negative_votes** (*int*) – The number of thumbs-down the game has received by users
- **game_mode** (*int*) – Game mode
- **engine** (*int*) – Source 1/Source 2
- **radiant_score** (*int*) – TODO
- **dire_score** (*int*) – TODO
- **flags** – TODO

has_leavers()

Returns `True` if the game contains a leaver

Return type bool

leavers()

Returns List of leavers in a game.

Return type list(*SteamAccount*)

class d2api.src.wrappers.**PlayerUnit**(*default_obj*)

An inventoried hero unit

Variables

- **steam_account** (SteamAccount) – Steam account of player
- **side** (*str*) – Side to which a player belongs (radiant/dire)
- **hero** (Hero) – Hero played
- **kills** (*int*) – Number of kills at the end of the match

- **deaths** (*int*) – Number of deaths at the end of the match
- **assists** (*int*) – Number of assists at the end of the match
- **leaver_status** (*int*) – Type of leaver
- **gold** (*int*) – Amount of gold remaining at the end of the match
- **last_hits** (*int*) – Number of list hits at the end of the match
- **denies** (*int*) – Number of denies at the end of the game
- **gold_per_minute** (*int*) – Overall gold/minute
- **xp_per_minute** (*int*) – Overall XP/min
- **gold_spent** (*int*) – Amount of gold spent during the match
- **hero_damage** (*int*) – Total damage done to other heroes at the end of the match
- **tower_damage** (*int*) – Total damage done to opponent towers at the end of the match
- **hero_healing** (*int*) – Total healing done to other heroes at the end of the match
- **additional_units** (*list (AdditionalUnit)*) – Additional units belonging to the current unit
- **inventory** (*list (Item)*) – List of inventory items
- **backpack** (*list (Item)*) – List of backpack items
- **ability_upgrades** (*list (AbilityInfo)*) – Ability upgrade information

all_items()

Returns Combined list of inventory and backpack items

Return type list(*Item*)

class d2api.src.wrappers.**AdditionalUnit**(*default_obj*)
An inventoried unit besides heroes (e.g. Lone druid bear)

Variables

- **inventory** (*list (Item)*) – List of inventory items
- **backpack** (*list (Item)*) – List of backpack items

all_items()

Returns Combined list of inventory and backpack items

Return type list(*Item*)

class d2api.src.wrappers.**PickBan**(*default_obj*)
Reprents a pick/ban during a game

Variables

- **is_pick** (*bool*) – `True` if the hero was picked
- **hero** (Hero) – Hero being picked/banned
- **side** (*str*) – Side that picked/banned this hero (radiant/dire)
- **order** (*int*) – Order in which the hero was picked/banned

5.3 get_heroes()

class `d2api.src.wrappers.`**`Heroes`**(*response_text*)

get_heroes response object

Variables

- **heroes** (*list (LocalizedHero)*) – List of localized hero information
- **count** (*int*) – Number of heroes returned

class `d2api.src.wrappers.`**`LocalizedHero`**(*default_obj*)

Localized hero information

Variables

- **name** (*str*) – Hero name
- **id** (*int*) – Hero ID
- **localized_name** (*str*) – Name of hero in language specified

5.4 get_game_items()

class `d2api.src.wrappers.`**`GameItems`**(*response_text*)

get_game_items response object

Variables **game_items** (*list (LocalizedGameItems)*) – List of localized item information

class `d2api.src.wrappers.`**`LocalizedGameItem`**(*default_obj*)

Localized item information

Variables

- **id** (*int*) – Item ID
- **name** (*str*) – Item name
- **cost** (*int*) – Cost of item
- **secret_shop** (*bool*) – True if the item is sold in secret shop
- **side_shop** (*bool*) – True if the item is sold in side shop
- **recipe** (*bool*) – True if it is a recipe
- **localized_name** (*str*) – Name of item in language specified

5.5 get_tournament_prize_pool()

class `d2api.src.wrappers.`**`TournamentPrizePool`**(*response_text*)

get_tournament_prize_pool response object

Variables

- **prize_pool** (*int*) – Prize pool
- **league_id** (*int*) – League ID for which prize pool was fetched

5.6 get_live_league_games()

class d2api.src.wrappers.**LiveLeagueGames**(*response_text*)

get_live_league_games response object

Variables **games** (*list (Game)*) – List of games

class d2api.src.wrappers.**Game**(*default_obj*)

Summary of a live league game

Variables

- **radiant_team** (TeamInfo) – Radiant team information
- **dire_team** (TeamInfo) – Dire team information
- **players** (*List (PlayerMinimal)*) – List of players in the game
- **scoreboard** (Scoreboard) – Game scoreboard at time of query
- **lobby_id** (*int*) – ID of lobby
- **match_id** (*int*) – Unique ID used to identify match
- **spectators** (*int*) – Number of spectators
- **league_id** (*int*) – Unique ID for the league of the match
- **league_node_id** (*int*) – Unique ID of node within the league
- **stream_delay_s** (*int*) – Stream delay in seconds
- **radiant_series_win** (*int*) – Number of wins by radiant team
- **dire_series_win** (*int*) – Number of wins by dire team
- **series_type** (*int*) – Type of series

class d2api.src.wrappers.**Scoreboard**(*default_obj*)

Scoreboard of live game

Variables

- **duration** (*int*) – Duration of the game at time of query
- **roshan_respawn_timer** (*int*) – Time left for Roshan to respawn
- **radiant** (TeamLive) – Radiant team summary
- **dire** (TeamLive) – Dire team summary

class d2api.src.wrappers.**TeamLive**(*default_obj*)

Information of a team in live game

Variables

- **score** (*int*) – Current number of kills by the team
- **buildings** (Buildings) – State of buildings
- **picks** (*list (Hero)*) – List of heroes picked
- **bans** (*list (Hero)*) – List of heroes banned
- **players** (*list (PlayerLive)*) – List of player summaries

class d2api.src.wrappers.**PlayerLive**(*default_obj*)

Information of a player in live game

Variables

- **player_slot** (*int*) – Slot of player within the team
- **steam_account** (SteamAccount) – Steam account of the player
- **hero** (Hero) – Hero played
- **kills** (*int*) – Number of kills
- **deaths** (*int*) – Number of deaths
- **assists** (*int*) – Number of assists
- **last_hits** (*int*) – Number of last hits
- **denies** (*int*) – Number of denies
- **gold** (*int*) – Current amount of gold
- **level** (*int*) – Current level
- **gold_per_min** (*int*) – gold/min at time of query
- **xp_per_min** (*int*) – XP/min at time of query
- **abilities** (*list (*AbilityInfo*)*) – List of ability information
- **ultimate_state** (*int*) – Current state of ultimate
- **ultimate_cooldown** (*int*) – Remaining time for ultimate to come off cooldown
- **inventory** (*list (*Item*)*) – List of items in player inventory
- **respawn_timer** (*int*) – Remain time for player to respawn
- **position_x** (*float*) – X coordinate of hero
- **position_y** (*float*) – Y coordinate of hero
- **net_worth** (*int*) – Net worth of the hero

5.7 get_top_live_game()

class d2api.src.wrappers.**TopLiveGame**(*response_text*)

get_top_live_game response object

Variables **game_list** (*list (*LiveGameSummary*)*) – List of top live games

class d2api.src.wrappers.**LiveGameSummary**(*default_obj*)

Summary of a live game

Variables

- **players** (PlayerMinimal) – List of player info
- **radiant_towers** (Buildings) – Radiant towers
- **dire_towers** (Buildings) – Dire towers
- **activate_time** (*int*) – TODO
- **deactivate_time** (*int*) – TODO
- **server_steam_id** (*int*) – Steam ID of server
- **lobby_id** (*int*) – ID of lobby

- **league_id** (*int*) – Unique ID for the league of the match
- **lobby_type** (*int*) – Type of lobby
- **game_time** (*int*) – Game time
- **delay** (*int*) – Stream delay (game, spectator delay)
- **spectators** (*int*) – Current number of spectators
- **game_mode** (*int*) – Game mode of current game
- **average_mmr** (*int*) – Average MMR of the game
- **match_id** (*int*) – Unique ID used to identify match
- **series_id** (*int*) – Unique ID used to identify series
- **radiant_team** (TeamInfo) – Information about radiant team
- **dire_team** (TeamInfo) – Information about dire team
- **sort_score** (*int*) – TODO
- **last_update_time** (*int*) – TODO
- **radiant_lead** (*int*) – Gold lead of radiant team
- **radiant_score** (*int*) – TODO
- **dire_score** (*int*) – TODO

5.8 get_team_info_by_team_id()

class d2api.src.wrappers.**TeamInfoByTeamID**(*response_text*)

get_team_info_by_team_id response object

Variables **teams** (*list (*TeamInfo*)*) – List of team information

5.9 get_broadcaster_info()

class d2api.src.wrappers.**BroadcasterInfo**(*response_text*)

get_broadcaster_info response object

Variables

- **steam_account** (SteamAccount) – Steam account of broadcaster
- **server_steam_id** (*int*) – Unique ID of game server currently being broadcasted
- **live** (*bool*) – True if the user is currently broadcasting
- **allow_live_video** (*bool*) – True if the user has allowed live video

5.10 get_player_summaries()

class d2api.src.wrappers.**PlayerSummaries**(*response_text*)

get_player_summaries response object

Variables **players** (*list (SteamDetails)*) – List of steam information in ascending order of account ids

class d2api.src.wrappers.**SteamDetails**(*default_obj*)
Information about a player as on Steam.

Variables

- **steam_account** (*SteamAccount*) – Steam account of the player
- **communityvisibility** (*str*) – A string representing the access setting of the profile
- **profilestate** (*int*) – Set to 1 if the user has configured their profile
- **personname** (*str*) – Display name
- **lastlogoff** (*int*) – Unix timestamp of when the player was last online
- **profileurl** (*str*) – The URL to the user's steam profile
- **avatar** (*str*) – URL of 32x32 image
- **avatarmedium** (*str*) – URL of 64x64 image
- **avatarfull** (*str*) – URL of 184x184 image
- **personastate** (*str*) – A string representing user's status
- **commentpermission** (*int*) – If present the profile allows public comments
- **realname** (*str*) – The user's real name
- **primaryclanid** (*int*) – The 64 bit ID of the user's primary group
- **timecreated** (*int*) – A unix timestamp of the date the profile was created
- **loccountrycode** (*int*) – ISO 3166 code of where the user is located
- **locstatecode** (*int*) – Variable length code representing the state the user is located in
- **loccityid** (*int*) – An integer ID internal to Steam representing the user's city
- **gameid** (*int*) – If the user is in game this will be set to it's app ID as a string
- **gameextrainfo** (*str*) – The title of the game
- **gameserverip** (*str*) – The server URL given as an IP address and port number

5.11 Common wrappers and entities

class d2api.src.wrappers.**TeamInfo**(*default_obj*)
Information about team

Variables

- **team_name** (*str*) – The team's name.
- **team_id** (*int*) – The team's unique ID.
- **team_logo** (*int*) – The UGC id for the team logo.
- **complete** (*bool*) – Whether the players for this team are all team members.

class d2api.src.wrappers.**AbilityInfo**(*default_obj*)
Ability upgrade during game.

Variables

- **ability** (Ability) – Ability upgraded.
- **time** (*int*) – Game time at which ability was upgraded
- **level** (*int*) – Level of the player at which ability was upgraded.

class d2api.src.wrappers.**Buildings**(*default_obj*)
Represents current state of buildings

Variables

- **{lane}_{position}** (*bool*) – Tower status [lane = top, mid, bot][position = 1, 2, 3] (e.g. top_t2)
- **ancient_bot** (*bool*) – Ancient bottom tower
- **ancient_top** (*bool*) – Ancient top tower
- **{lane}_{type}** (*bool*) – Barracks status [lane = top, mid, bot][type = ranged, melee] (e.g. mid_melee)

class d2api.src.wrappers.**PlayerMinimal**(*default_obj*)
A minimal information wrapper for a player

Variables

- **steam_account** (SteamAccount) – Steam account of player
- **side** (*str*) – side to which a player belongs (radiant/dire)
- **hero** (Hero) – hero played

class d2api.src.entities.**Ability**(*ability_id*)
Wrapper to map ability data to ability_id

Variables

- **ability_id** (*int*) – Unique identifier of ability
- **ability_name** (*str*) – Name of the ability

class d2api.src.entities.**Item**(*item_id*)
Wrapper to map item information to item_id

Variables

- **item_id** (*int*) – Unique identifier of item
- **item_cost** (*int*) – Cost of the item
- **item_aliases** (*list (str)*) – List of names by which the item is known
- **item_name** (*str*) – Name of the item

class d2api.src.entities.**SteamAccount**(*account_id*)
Wrapper to implicitly store steam32 and steam64 account IDs

Variables

- **id32** (*int*) – 32-bit Steam ID
- **id64** (*int*) – 64-bit Steam ID

class d2api.src.entities.**Hero**(*hero_id*)
Wrapper to map hero information to hero_id

Variables

- **hero_id** (*int*) – Unique identifier of hero

- **`hero_name`** (*str*) – Name of the hero

Python Module Index

Index