
PyVainglory Documentation

Release 0.5.17

PyVainglory

Jan 04, 2018

Contents

1	pyvainglory.Client	3
2	pyvainglory.AsyncClient	7
3	pyvainglory.models	11
4	pyvainglory.errors	17
	Python Module Index	19

Basic Usage:

```
import pyvainglory
import requests

vgc = pyvainglory.Client('your-api-key')

# You can also provide a requests.Session to the Client constructor
session = requests.Session()
vgc_a = pyvainglory.Client('your-api-key', session)

# Get 3 matches after specified time
# after and before can also be datetime.datetime objects
matches = vgc.get_matches(limit=3, after="2017-11-22T20:34:58Z", region='na')

# Go to the next pages of matches
matches.next()

# Get telemetry data for one of the matches
telemetry = matches.matches[0].get_telemetry()

player = vgc.player_by_name('Demolasher36', region='sg')
my_blitz_games = player.games_played.blitz

my_recent_games = vgc.get_matches(limit=3, after="2018-01-1T20:34:58Z", playernames=[
    ↪ 'Demolasher36'], region='sg')
```

Async Usage:

```
import aiohttp
import asyncio
import pyvainglory

vgc = pyvainglory.AsyncClient('your-api-key')

# You can also provide an aiohttp.ClientSession to the AsyncClient constructor
session = aiohttp.ClientSession()
vgc_a = pyvainglory.AsyncClient('your-api-key', session)

# Get 3 matches after specified time
# after and before can also be datetime.datetime objects
matches = await vgc.get_matches(limit=3, after="2017-11-22T20:34:58Z", region='na')

# Go to the next pages of matches
await matches.next()

# Get telemetry data for one of the matches
telemetry = await matches.matches[0].get_telemetry()

player = await vgc.player_by_name('Demolasher36', region='sg')
my_blitz_games = player.games_played.blitz

my_recent_games = await vgc.get_matches(limit=3, after="2018-01-1T20:34:58Z", ↵
    ↪ playernames=['Demolasher36'], region='sg')
```


class pyvainglory.client.**Client** (*key, session: requests.sessions.Session = None*)

Bases: pyvainglory.clientbase.ClientBase

Top level class for user to interact with the API.

Parameters **key** : str

The official Vainglory API key.

session : Optional[requests.Session]

get_matches (*offset: int = None, limit: int = None, after=None, before=None, playerids: list = None, playernames: list = None, gamemodes: list = None, region: str = None*)

Access the /matches endpoint and grab a list of matches

Parameters **offset** : Optional[int]

The nth number of match to start the page from.

limit : Optional[int]

Number of matches to return.

after : Optional[str or datetime.datetime]

Filter to return matches after provided time period, if an str is provided it should follow the **iso8601** format.

before : Optional[str or datetime.datetime]

Filter to return matches before provided time period, if an str is provided it should follow the **iso8601** format.

playerids : list(str)

Filter to only return matches with provided players in them by looking for their player IDs.

playernames : list(str)

Filter to only return matches with provided players in them by looking for their player-names.

gamemodes : list(str)

Filter to to return only matches that match with the gamemodes in the provided list.

region : str

The region to look for matches in.

Returns `pyvainglory.models.MatchPaginator`

A MatchPaginator instance representing a get_matches request

get_players (*region: str, playerids: list = None, usernames: list = None*)

Get multiple players' info at once.

Parameters **playerids** : list(str)

A list of playerids, either a *list* of strs or a *list* of ints. Max list length is 6.

usernames : list(str)

A list of usernames, a *list* of strings, case sensitive. Max list length is 6

region : str

The region to look for players in.

Returns list

A list of `pyvainglory.models.Player`

get_status ()

Check if the API is up and running

Returns tuple(createdAt: str, version: str):

match_by_id (*match_id, region: str = None*)

Get a Match by its ID.

Parameters **match_id** : str

region : str

The region to look for this match in.

Returns `pyvainglory.models.Match`

A match object representing the requested match.

player_by_id (*player_id: int, region: str*)

Get a player's info by their ID.

Parameters **player_id** : int

region : str

The region to look for this match in.

Returns `pyvainglory.models.Player`

A Player object representing the requested player.

player_by_name (*username: str, region: str*)

Get a player's info by their ingame name.

Parameters **username** : str

Case insensitive username

region : str

The region to look for this player in.

Returns *pyvainglory.models.Player*

A Player object representing the requested player.

pyvainglory.AsyncClient

```
class pyvainglory.asyncclient.AsyncClient (key, session: aiohttp.client.ClientSession =  
                                           None)  
    Bases: pyvainglory.clientbase.ClientBase
```

Top level class for user to interact with the API.

Parameters **key** : str

The official Vainglory API key.

session : Optional[requests.Session]

```
get_matches (offset: int = None, limit: int = None, after=None, before=None, playerids: list = None,  
              playernames: list = None, gamemodes: list = None, region: str = None)
```

Access the /matches endpoint and grab a list of matches

Parameters **offset** : Optional[int]

The nth number of match to start the page from.

limit : Optional[int]

Number of matches to return.

after : Optional[str or datetime.datetime]

Filter to return matches after provided time period, if an str is provided it should follow the **iso8601** format.

before : Optional[str or datetime.datetime]

Filter to return matches before provided time period, if an str is provided it should follow the **iso8601** format.

playerids : list(str)

Filter to only return matches with provided players in them by looking for their player IDs.

playernames : list(str)

Filter to only return matches with provided players in them by looking for their player-names.

gamemodes : list(str)

Filter to to return only matches that match with the gamemodes in the provided list.

region : str

The region to look for matches in.

Returns `pyvainglory.models.MatchPaginator`

A MatchPaginator instance representing a get_matches request

get_players (*region: str, playerids: list = None, usernames: list = None*)

Get multiple players' info at once.

Parameters **playerids** : list(str)

A list of playerids, either a *list* of strs or a *list* of ints. Max list length is 6.

usernames : list(str)

A list of usernames, a *list* of strings, case sensitive. Max list length is 6

region : str

The region to look for players in.

Returns list

A list of `pyvainglory.models.Player`

get_status ()

Check if the API is up and running

Returns tuple(createdAt: str, version: str):

match_by_id (*match_id, region: str = None*)

Get a Match by its ID.

Parameters **match_id** : str

region : str

The region to look for this match in.

Returns `pyvainglory.models.Match`

A match object representing the requested match.

player_by_id (*player_id: int, region: str*)

Get a player's info by their ID.

Parameters **player_id** : int

region : str

The region to look for this match in.

Returns `pyvainglory.models.Player`

A Player object representing the requested player.

player_by_name (*username: str, region: str*)

Get a player's info by their ingame name.

Parameters **username** : str

Case insensitive username

region : str

The region to look for this player in.

Returns *pyvainglory.models.Player*

A Player object representing the requested player.

class pyvainglory.models.**AsyncMatch** (*data, session, included=None*)

Bases: *pyvainglory.models.MatchBase*

Extends *MatchBase* to add async *get_telemetry()*.

get_telemetry (*session=None*)

Get telemetry data for a match.

Parameters *session* : Optional[*aiohttp.ClientSession*]

Optional session to use to request telemetry data.

Returns *dict*

Match telemetry data

class pyvainglory.models.**AsyncMatchPaginator** (*matches, data, client*)

Bases: *pyvainglory.models.Paginator*

Returned only by the *get_matches* method of the async client.

Attributes

matches	(list(<i>AsyncMatch</i>)) A list of matches.
----------------	--

first (*session=None*)

Move to the first page of matches.

Parameters *session* : Optional[*aiohttp.ClientSession*]

Optional session to use to make this request.

Returns *list*

A list of *Match*.

next (*session=None*)

Move to the next page of matches.

Parameters *session* : Optional[\[aiohttp.ClientSession\]](#)

Optional session to use to make this request.

Returns *list*

A list of *Match*.

Raises *VGPaginationError*

The current page is the last page of results

prev (*session=None*)

Move to the previous page of matches.

Parameters *session* : Optional[\[aiohttp.ClientSession\]](#)

Optional session to use to make this request.

Returns *list*

A list of *Match*.

Raises *VGPaginationError*

The current page is the first page of results

class `pyvainglory.models.BaseVGObject` (*data*)

Bases: `object`

A base object for most data classes

Attributes

id	(str) A general unique ID for each type of data.
-----------	--

class `pyvainglory.models.Match` (*data, session, included=None*)

Bases: `pyvainglory.models.MatchBase`

Extends *MatchBase* to add *get_telemetry()*

get_telemetry (*session=None*)

Get telemetry data for a match.

Parameters *session* : Optional[\[requests.Session\]](#)

Optional session to use to request telemetry data.

Returns *dict*

Match telemetry data

class `pyvainglory.models.MatchBase` (*data, session, included=None*)

Bases: `pyvainglory.models.BaseVGObject`

A class that holds data for a match.

Attributes

id	(str) A general unique ID for each type of data.
created_at	(<code>datetime.datetime</code>) Time when the match was created.
duration	(int) The match's duration in seconds.
game_mode	(tuple(mode_code: str, friendly_mode_name: str)) The gamemode this match was of.
patch	(str) The Vainglory patch version this match was played on.
region	(tuple(region_code: str, friendly_region_name: str)) The region shard on which this match data resides.
game_end_reason	(str) Can either be 'victory' or 'surrender'.
rosters	(list(<i>Roster</i>)) A list of <i>Roster</i> objects representing rosters taking part in the match.
spectators	(list(<i>Participant</i>)) A list of <i>Participant</i> objects representing spectators.
telemetry_url	(str) URL for the telemetry file for this match
session	(<code>aiohttp.ClientSession</code> or <code>requests.Session</code>) Depends on which class extends MatchBase, AsyncClient or normal Client, respectively.

class `pyvainglory.models.MatchPaginator` (*matches*, *data*, *client*)

Bases: `pyvainglory.models.Paginator`

Returned only by the `get_matches` method of the client.

Attributes

matches	(list(<i>Match</i>)) A list of matches.
----------------	---

first (*session=None*)

Move to the first page of matches.

Parameters *session* : Optional[`requests.Session`]

Optional session to use to make this request.

Returns *list*

A list of *Match*.

next (*session=None*)

Move to the next page of matches.

Parameters *session* : Optional[`requests.Session`]

Optional session to use to make this request.

Returns *list*

A list of *Match*.

Raises `VGPaginationError`

The current page is the last page of results

prev (*session=None*)

Move to the previous page of matches.

Parameters *session* : Optional[`requests.Session`]

Optional session to use to make this request.

Returns *list*

A list of *Match*.

Raises `VGPageinationError`

The current page is the first page of results

class `pyvainglory.models.Paginator(matches, data, client)`
Bases: `object`

Returned only by pyvainglory's client classes.

class `pyvainglory.models.Participant(participant, included)`
Bases: `pyvainglory.models.BaseVGObject`

A class that holds data about a participant in a match.

Attributes

id	(str) A general unique ID for each type of data.
actor	(str)
region	(tuple(region_code: str, friendly_region_name: str))
assists	(int)
crystal_mines_captured	(int)
deaths	(int)
farm	(int)
first_time_afk	(bool)
gold	(int)
gold_mines_captured	(int)
items_bought	(dict())
items_sold	(dict())
items_used	(dict())
final_build	(list(str))
jungle_kills	(int)
kills	(int)
krakens_captured	(int)
minion_kills	(int)
skin	(str)
turrets_captured	(int)
player	(<i>Player</i>)

class `pyvainglory.models.Player(data)`
Bases: `pyvainglory.models.BaseVGObject`

A class that holds general user data, if this is through a Match, this will not have name, picture and title, only an ID

Attributes

id	(str) A general unique ID for each type of data.
name	(str) The player's name
region	(tuple(region_code: str, friendly_region_name: str)) The region id for this Player's region, ex: 'sg', 'na', etc.
elo_history	(dict) The player's past highest elo for seasons 4 through 9.
karma_level	(tuple(karma: int, text_karma: str)) The karma level for this player, the tuple provides a text translation for each karma level, i.e 0, 1 and 2.
account_level	(int) The Player's account's level.
life-time_gold	(int) Total amount of gold this player has earned since the account was created.
games_played	(namedtuple('games_played', 'battle_royale blitz casual ranked onslaught')) The attributes can be accessed using dotted access, ex: games_played.battle_royale.
skill_tier	(int) The Player's account's current skill tier.
win_streak	(int) The Player's current win streak.
wins	(int) Total amount of wins the player has had since the account was created.
xp	(int) The account's XP level.

class pyvainglory.models.**Roster** (*roster, included*)

Bases: *pyvainglory.models.BaseVGObject*

A class that holds data about one of the two teams in a match.

Attributes

id	(str) A general unique ID for each type of data.
region	(tuple(region_code: str, friendly_region_name: str)) The region shard on which this roster data resides.
aces	(int)
gold	(int)
hero_kills	(int)
krakens_captured	(int)
side	(str) This is in the format of 'side/color', ex: 'left/blue'.
turret_kills	(int)
turrets_remaining	(int)
won	(bool) Signifies if the roster won a match.
participants	(list(<i>Participant</i>)) A list of <i>Participant</i> representing players that are part of the roster.

class pyvainglory.models.**current_elo** (*blitz, ranked*)

Bases: tuple

blitz

Alias for field number 0

ranked

Alias for field number 1

```
class pyvainglory.models.games_played(battle_royale, blitz, casual, ranked, onslaught)
    Bases: tuple

    battle_royale
        Alias for field number 0

    blitz
        Alias for field number 1

    casual
        Alias for field number 2

    onslaught
        Alias for field number 4

    ranked
        Alias for field number 3
```

CHAPTER 4

pyvainglory.errors

exception `pyvainglory.errors.EmptyResponseException (error)`

Bases: `Exception`

Raised when any request is 200 OK, but the data is empty.

exception `pyvainglory.errors.NotFoundException (response, data)`

Bases: `pyvainglory.errors.VGRequestException`

For the 404s

exception `pyvainglory.errors.VGFilterException (error)`

Bases: `Exception`

Raised when an invalid filter value is supplied.

exception `pyvainglory.errors.VGPaginationError (error)`

Bases: `Exception`

Raised when `pyvainglory.models.MatchPaginator.next()` or `pyvainglory.models.MatchPaginator.prev()` are called when the paginator is on the last or first page respectively.

exception `pyvainglory.errors.VGRequestException (response, data)`

Bases: `Exception`

General purpose exception, base class for other request related exceptions.

Parameters `response` : `aiohttp.ClientResponse`

`data` : dict

The json response from the API

exception `pyvainglory.errors.VGServerException (response, data)`

Bases: `pyvainglory.errors.VGRequestException`

Exception that signifies that the server failed to respond with valid data.

p

`pyvainglory.asyncclient`, [7](#)

`pyvainglory.client`, [3](#)

`pyvainglory.errors`, [17](#)

`pyvainglory.models`, [11](#)

A

AsyncClient (class in pyvainglory.asyncclient), 7
AsyncMatch (class in pyvainglory.models), 11
AsyncMatchPaginator (class in pyvainglory.models), 11

B

BaseVGObject (class in pyvainglory.models), 12
battle_royale (pyvainglory.models.games_played attribute), 16
blitz (pyvainglory.models.current_elo attribute), 15
blitz (pyvainglory.models.games_played attribute), 16

C

casual (pyvainglory.models.games_played attribute), 16
Client (class in pyvainglory.client), 3
current_elo (class in pyvainglory.models), 15

E

EmptyResponseException, 17

F

first() (pyvainglory.models.AsyncMatchPaginator method), 11
first() (pyvainglory.models.MatchPaginator method), 13

G

games_played (class in pyvainglory.models), 15
get_matches() (pyvainglory.asyncclient.AsyncClient method), 7
get_matches() (pyvainglory.client.Client method), 3
get_players() (pyvainglory.asyncclient.AsyncClient method), 8
get_players() (pyvainglory.client.Client method), 4
get_status() (pyvainglory.asyncclient.AsyncClient method), 8
get_status() (pyvainglory.client.Client method), 4
get_telemetry() (pyvainglory.models.AsyncMatch method), 11
get_telemetry() (pyvainglory.models.Match method), 12

M

Match (class in pyvainglory.models), 12
match_by_id() (pyvainglory.asyncclient.AsyncClient method), 8
match_by_id() (pyvainglory.client.Client method), 4
MatchBase (class in pyvainglory.models), 12
MatchPaginator (class in pyvainglory.models), 13

N

next() (pyvainglory.models.AsyncMatchPaginator method), 11
next() (pyvainglory.models.MatchPaginator method), 13
NotFoundException, 17

O

onslaught (pyvainglory.models.games_played attribute), 16

P

Paginator (class in pyvainglory.models), 14
Participant (class in pyvainglory.models), 14
Player (class in pyvainglory.models), 14
player_by_id() (pyvainglory.asyncclient.AsyncClient method), 8
player_by_id() (pyvainglory.client.Client method), 4
player_by_name() (pyvainglory.asyncclient.AsyncClient method), 8
player_by_name() (pyvainglory.client.Client method), 4
prev() (pyvainglory.models.AsyncMatchPaginator method), 12
prev() (pyvainglory.models.MatchPaginator method), 13
pyvainglory.asyncclient (module), 7
pyvainglory.client (module), 3
pyvainglory.errors (module), 17
pyvainglory.models (module), 11

R

ranked (pyvainglory.models.current_elo attribute), 15
ranked (pyvainglory.models.games_played attribute), 16

Roster (class in `pyvainglory.models`), [15](#)

V

`VGFilterException`, [17](#)

`VGPaginationError`, [17](#)

`VGRequestException`, [17](#)

`VGServerException`, [17](#)