steampak Documentation

Release 0.5.0

Igor ‘idle sign‘ Starikov

Mar 05, 2017

Contents

https://github.com/idlesign/steampak

This software is not affiliated with Valve, Steam, or any of their partners.

CHAPTER 1

Description

Nicely packed tools to work with Steam APIs

- Steam library bindings for Python programming language.

  It allows your game to interact with features offered by Steam client and Steam platform.

- Tools for querying Steam Web resources.

  Allowing access to applications and market data from within your Python application.

- Command line utility.

  To reach and analyse publicly available applications and market information.

CHAPTER 2

Requirements

- Python 2.7+, 3.3+
- Steam API library 1.32+ from Steamworks SDK
- Linux (not tested with libraries for OSX or Windows)

CHAPTER 3

Optional dependencies

To install all dependencies for *steampak*:

```
> pip install steampak[extra]
```

- *requests*, *BeautifulSoup* (for Web API related stuff)
- *click* (for CLI)

CHAPTER 4

Table of Contents

Quickstart

Steam API example

- First a path to steam library file. This library should be provided with your game. Library for various OS (*libsteam_api.so* or *steam_api.dll* or *libsteam_api.dylib*) is distributed with Steam SDK available for Steam partners at https://partner.steamgames.com/
- Second. Your need to know application (game) identifier provided by Steam. Pass it as a parameter or put *steam_appid.txt* file with that ID in your game folder.

Now you're ready to begin. API initialization is easy:

```
from steampak import SteamApi  # Main API entry point.

LIBRARY_PATH = '/home/me/my_steam_game/libsteam_api.so'
APP_ID = 480  # We use `Spacewar` ID. (This game is provided with SDK).

api = SteamApi(LIBRARY_PATH, app_id=APP_ID)
```

After that you can access various API parts as *api* object attributes:

```
# Let's print some friend names:
for user in api.friends():
    print(user.name)

# Print out some info from utils:
print(api.utils.country_code)
print(api.utils.ui_language)

# Achievements progress:
for ach_name, ach in api.apps.current.achievements():
    print('%s (%s): %s' % (ach.title, ach_name, ach.get_unlock_info()))
```

```
# Installed applications titles:
for app_id, app in api.apps.installed():
    print('%s: %s' % (app_id, app.name))
```

When you're done, do not forget to shutdown API:

```
# Do not forget to shutdown when done:
api.shutdown()
```

Command line interface example

```
; Get prices and simple analysis for Half-Life 2 cards.
$ steampak app 220 get_card_prices --currency USD

; Get `Gordon Freeman` card price.
$ steampak market item 220 "Gordon Freeman" get_price --currency GBP

; Get games owned by `idlesign`.
$ steampak user idlesign get_games
```

Use `--help` command option to get more information on available commands.

Applications

class `steampak.libsteam.resources.apps.`**`Applications`**

Exposes methods to get applications data.

`current` **= <steampak.libsteam.resources.apps.CurrentApplication object>**

Interface to current application.

```
print(api.apps.current.language_current)
```

`installed` **= <steampak.libsteam.resources.apps.InstalledApplications object>**

Interface to installed applications.

```
for app_id, app in api.apps.installed():
    print('%s: %s' % (app_id, app.name))
```

Current Application

class `steampak.libsteam.resources.apps.`**`CurrentApplication`**

Exposes methods to get current application data.

Interface can be accessed through `api.apps.current`.

```
from steampak import SteamApi

api = SteamApi(LIBRARY_PATH, app_id=APP_ID)

print(api.apps.current.language_current)
```

`achievements` = <steampak.libsteam.resources.stats.CurrentApplicationAchievements object>

Current application (game) achievements.

```
for ach_name, ach in api.apps.current.achievements():
    print('%s: %s' % (ach_name, ach.title))
```

`beta_name`

Current beta branch name, 'public' is the default branch.

Return type str

`build_id`

Current application Build ID. This may change at any time based on backend updates.

Warning: Restricted interface can only be used by approved apps.

Return type int

`dlcs` = <steampak.libsteam.resources.apps.CurrentApplicationDlcs object>

Interface to DLCs of current application.

```
for dlc_id, dlc in api.apps.current.dlcs():
    print('%s: %s' % (dlc_id, dlc.name))
```

`install_dir`

Returns application installation path.

Note: If fails this falls back to a restricted interface, which can only be used by approved apps.

Return type str

`installed`

`True` if app is installed (not necessarily owned).

Return type bool

`language_available`

List of available game languages.

E.g.: ['english', 'russian']

Return type list[str]

`language_current`

Current game language.

E.g.: english

Return type str

`low_violence`

`True` if the current app is low violence.

Return type bool

`mark_corrupt`(*only_files_missing=False*)

Signal Steam that game files seems corrupt or missing.

Parameters **only_files_missing** (*bool*) – Set it to True if only files are missing.

Return type bool

mode_cybercafe

True if the current app supports Valve Cybercafe Program.

Return type bool

mode_free_weekend

True if the user is subscribed to the current app through a free weekend.

Will return False for users who have a retail or other type of license.

Note: Before using, please ask your Valve technical contact how to package and secure your free weekened.

Return type bool

name

Application name, or None on error.

Warning: Restricted interface can only be used by approved apps.

Return type str

owned

True if user owns the current app.

Return type bool

owner

Owner user. If different from current user, app is borrowed.

Return type *User*

purchase_time

Date and time of app purchase.

Return type datetime

vac_banned

True if the current app is banned by BIsVACBanned.

Return type bool

Application

steampak.**SteamApplication**

alias of Application

class steampak.libsteam.resources.apps.**Application**(*app_id*)

Exposes methods to get application data.

Aliased as steampak.SteamApplication.

```
from steampak import SteamApplication

# We use `Spacewar` app ID. (This game is provided with SDK).
my_app = SteamApplication(480)
```

Parameters `app_id` (*int* | *None*) – Application (game) ID.

`build_id`

Application Build ID. This may change at any time based on backend updates.

> **Warning:** Restricted interface can only be used by approved apps.

Return type int

`install_dir`

Returns application installation path.

> **Note:** If fails this falls back to a restricted interface, which can only be used by approved apps.

Return type str

`installed`

`True` if app is installed (not necessarily owned).

Return type bool

`name`

Application name, or None on error.

> **Warning:** Restricted interface can only be used by approved apps.

Return type str

`owned`

`True` if user owns the current app.

> **Warning:** Only use this member if you need to check ownership of a game related to yours, a demo for example.

Return type bool

`purchase_time`

Date and time of app purchase.

Return type datetime

DLC

`steampak.SteamDlc`

alias of `Dlc`

class `steampak.libsteam.resources.apps.Dlc`(*app_id*)

Exposes methods to get downloadable content (DLC) data.

Aliased as `steampak.SteamDlc`.

```
from steampak import SeamDlc

# We use `Spacewar` DLC app ID. (Spacewar game is provided with SDK).
my_dlc = SeamDlc(110902)
```

Current application DLCs are available through `CurrentApplication.dlcs`.

available

True if DLC is available.

Return type bool

build_id

Application Build ID. This may change at any time based on backend updates.

Warning: Restricted interface can only be used by approved apps.

Return type int

get_download_progress()

Returns tuple with download progress (for optional DLCs):

(bytes_downloaded, bytes_total)

Return type tuple

install()

Installs DLC (for optional DLCs).

install_dir

Returns application installation path.

Note: If fails this falls back to a restricted interface, which can only be used by approved apps.

Return type str

installed

`True` if the user owns the DLC & if the DLC is installed.

Return type bool

name

DLC name.

Return type str

owned

`True` if user owns the current app.

> **Warning:** Only use this member if you need to check ownership of a game related to yours, a demo for example.

Return type bool

purchase_time

Date and time of app purchase.

Return type datetime

uninstall()

Uninstalls DLC (for optional DLCs).

Steam API (Main Entry Point)

steampak.**SteamApi**

alias of `Api`

class steampak.libsteam.resources.main.**Api**(*library_path*, *app_id=None*)

Main entry point of Steam API.

It is aliased as `steampak.SteamApi`.

```
from steampak import SteamApi

# Automatically initialize Steam API library if Steam client is running.
api = SteamApi(LIBRARY_PATH, app_id=APP_ID)

# Do not forget to shutdown when done:
api.shutdown()
```

Parameters

- **library_path** (*str*) – Full path to Steam library file. The library should be provided with your game. Library for various OS is distributed with Steam SDK available for Steam partners at https://partner.steamgames.com/
- **app_id** (*str|int*) – Application (game) identifier. Pass it as a parameter or put *steam_appid.txt* file with that ID in your game folder.

app_id

Application ID of the current process.

apps = <steampak.libsteam.resources.apps.Applications object>

Interface to applications (games).

```
for app_id, app in api.apps.installed():
    print('%s: %s' % (app_id, app.name))
```

current_user = <steampak.libsteam.resources.user.CurrentUser object>

Interface to current user.

```
print(api.current_user.name)
```

friends = <steampak.libsteam.resources.friends.Friends object>

Interface to friends of current user.

```
for user in api.friends():
    print(user.name)
```

groups = <steampak.libsteam.resources.groups.Groups object>

Interface to user groups.

```
for group in api.groups():
    print(group.name)
```

init(*app_id=None*)

Initializes Steam API library.

Parameters **app_id** (*str | int*) – Application ID.

Raises SteamApiStartupError

install_path

Returns library installation path.

Return type str

overlay = <steampak.libsteam.resources.overlay.Overlay object>

Interface to Steam overlay.

Overlay-related functions only work with OpenGL/D3D applications and only if Steam API is initialized before renderer device.

```
api.overlay.activate()
```

classmethod set_app_id(*app_id*)

Sets current application ID into environment.

Parameters **app_id** (*str | int*) – Your application ID.

shutdown()

Shutdowns API.

start_app()

Detects if your executable was launched through the Steam client, and restarts your game through the client if necessary. The Steam client will be started if it is not running.

SDK Note: This function should be used only if you are using CEG or not using Steam's DRM. Once applied to your executable, Steam's DRM will handle restarting through Steam if necessary.

Return type bool

Returns

True if your executable was NOT launched through the Steam client. This function will then start your application through the client. Your current process should exit.

False if your executable was started through the Steam client or a steam_appid.txt file is present in your game's directory (for development). Your current process should continue.

steam_running

`True` if a local Steam client is running

Return type bool

utils = <steampak.libsteam.resources.utils.Utils object>
Interface to various utilities.

```
print(api.utils.ui_language)
```

Friends

class `steampak.libsteam.resources.friends.`**Friends**
Exposes methods to get friends related data.

Interface can be accessed through `api.friends()`:

```
for user in api.friends():
    print(user.name)
```

get_count (*flt=65535*)
Returns a number of current user friends, who meet a given criteria (filter).

Parameters **flt** (*int*) – Filter value from FriendFilter. Filters can be combined with |. Defaults to `FriendFilter.ALL`.

Return type int

tags = <steampak.libsteam.resources.friends.FriendTags object>
Interface to friend tags (categories).

```
for tag in api.friends.tags():
    print(tag.name)
```

Friend Tag

class `steampak.libsteam.resources.friends.`**FriendTag** (*tag_id*)
Exposes methods to get friend tag data.

Interface can be accessed through `api.friends.tags()`:

```
for tag in api.friends.tags():
    print(tag.name)
```

name
Name of a friend tag, or None on error.

Return type str

Friend Filter

class `steampak.libsteam.resources.base.`**FriendFilter**
Filters to be provided to functions returning friends. Can be combined using |.

ALL = 65535

BLOCKED = 1

CLAN_MEMBER = 8

FRIENDSHIP_REQUESTED = 2

FRIEND_OF_FRIEND = 64

HAS_PLAYED_WITH = 32

IGNORED = 512

IGNORED_FRIEND = 1024

IMMEDIATE = 4

NONE = 0

ON_GAME_SERVER = 16

REQUESTING_FRIENDSHIP = 128

REQUESTING_INFO = 256

SUGGESTED = 2048

Groups

class `steampak.libsteam.resources.groups.`**`Groups`**

Exposes methods to get user groups data. Groups are also known as clans.

Interface can be accessed through `api.groups()`:

```
for group in api.groups():
    print(group.name)
```

Group

class `steampak.libsteam.resources.groups.`**`Group`**(*group_id*)

Exposes methods to get user groups (clans) data.

Instances can be accessed through `api.groups()`:

```
for group in api.groups():
    print(group.name)
```

alias

Alias (short name) of a group.

Return type str

name

Name of a group.

Return type str

open_chat()

Shows overlay with group chat window.

show_page()

Shows overlay with group page.

stats

Basic group statistics.

Returned dict has the following keys:

‘online’ - users online count ‘ingame’ - users currently in game count ‘chatting’ - users chatting count

Returns dict

Statistics

Achievement

`steampak.`**`SteamAchievement`**

alias of `Achievement`

class `steampak.libsteam.resources.stats.`**`Achievement`**`(`*name*`)`

Exposes methods to get achievement data.

Aliased as `steampak.SteamAchievement`.

```
from steampak import SteamAchievement

print(SteamAchievement('some_achievement_name').title)
```

Instances can be accessed through `api.apps.current.achievements()`:

```
for ach_name, ach in api.apps.current.achievements():
    print('%s (%s)' % (ach.title, ach_name))
```

`clear``(`*store=True*`)`

Clears (locks) the achievement.

Return type bool

description

Achievement description.

Return type str

`get_unlock_info``()`

Returns tuple of unlock data: (is_unlocked, unlocked_datetime).

Note: *unlocked_datetime* will be `None` if achievement if unlocked before 2009-12-01.

Return type tuple[bool, datetime]

global_unlock_percent

Global achievement unlock percent.

Return type float

hidden

`True` if achievement is hidden.

Return type bool

title
Achievement title.

Return type str

unlock (*store=True*)
Unlocks the achievement.

Parameters **store** (*bool*) – Whether to send data to server immediately (as to get overlay notification).

Return type bool

unlocked
`True` if achievement is unlocked.

Return type bool

Users

Current User

class `steampak.libsteam.resources.user.`**CurrentUser**
Exposed methods related to a current Steam client user.

Can be accessed through `api.current_user`:

```
user = api.current_user
```

behind_nat
`True` if this users looks like they are behind a NAT device. Only valid once the user has connected to steam (i.e a SteamServersConnected_t has been issued) and may not catch all forms of NAT.

Return type bool

level
Current user level (as shown on their profile).

Return type int

logged_in
`True` if the Steam client current has a live connection to the Steam servers.

If `False`, it means there is no active connection due to either a networking issue on the local machine, or the Steam server is down/busy.

The Steam client will automatically be trying to recreate the connection as often as possible.

Return type bool

User

class `steampak.libsteam.resources.user.`**User** (*user_id*)
Exposes methods to get user-related data.

Instance access example:

```
for user in api.friends():
    print(user.name)
```

accept_friend_invite()
Shows a dialog to accept an incoming friend invite.

add_to_friends()
Shows a dialog to add user as a friend.

get_state(*as_str=False*)
Returns user state. See `UserState`.

Parameters **as_str** (*bool*) – Return human-friendly state name instead of an ID.

Return type int|str

has_friends(*flt=65535*)
Indicated whether the user has friends, who meet the given criteria (filter).

Parameters **flt** (*int*) – Filter value from FriendFilter. Filters can be combined with |.

Return type bool

ignore_friend_invite()
Shows a dialog to ignore an incoming friend invite.

level
User level (as shown on profile).

Return type int

name
User name (the same name as on the users community profile page).

Return type str

name_history
A list of user names (as user can change those occasionally).

Return type list

nickname
A nickname the current user has set for the user, or `None` if not set.

Return type str

open_chat()
Shows overlay with chat window.

remove_from_friends()
Shows a dialog to remove user from friends.

show_achievements()
Shows overlay with user achievements.

show_profile()
Shows overlay with user profile.

show_stats()
Shows overlay with user stats.

state
User state. See .get_state().

Return type str

User State

class `steampak.libsteam.resources.user.`**`UserState`**

User states enumeration.

`AWAY` = **3**

`BUSY` = **2**

`OFFLINE` = **0**

`ONLINE` = **1**

`READY_TO_PLAY` = **6**

`READY_TO_TRADE` = **5**

`SNOOZE` = **4**

`aliases` = **{0: ‘offline’, 1: ‘online’, 2: ‘busy’, 3: ‘away’, 4: ‘snooze’, 5: ‘trade’, 6: ‘play’}**

`get_alias`(*item_id*)

Returns item alias by ID.

Parameters **`item_id`** (*`int`*) – Item ID.

Return type str

Utils

class `steampak.libsteam.resources.utils.`**`Utils`**

Exposes various utility methods.

Interface can be accessed through `api.utils`:

```
print(api.utils.ui_language)
```

`app_id`

Application ID of the current process.

Return type int

`battery_power`

The amount of battery power left in the current system in % [0..100]. 255 for being on AC power.

Return type int

`country_code`

2 digit ISO 3166-1-alpha-2 format country code this client is running in (as looked up via an IP-to-location database)

E.g: RU.

Return type str

`get_universe`(*as_str=False*)

Returns universe the client is connected to. See `Universe`.

Parameters **`as_str`** (*`bool`*) – Return human-friendly universe name instead of an ID.

Return type int|str

ipc_call_count

The number of IPC calls made since the last time this function was called. Used for perf debugging so you can understand how many IPC calls your game makes per frame. Every IPC call is at minimum a thread context switch if not a process one so you want to rate control how often you do them.

Return type int

overlay_enabled

`True` if the overlay is running & the user can access it.

The overlay process could take a few seconds to start & hook the game process, so this function will initially return `False` while the overlay is loading.

Return type bool

seconds_app_active

Number seconds application is active.

Return type int

seconds_computer_active

Number seconds computer is active.

Return type int

server_time

Date and time on server.

Return type datetime

set_notification_position(*position*)

Sets the position where the overlay instance for the currently calling game should show notifications.

This position is per-game and if this function is called from outside of a game context it will do nothing.

Parameters **position** (*int*) – Position. See `NotificationPosition`.

ui_language

The language the steam client is running in.

E.g.: russian

Return type str

universe

Universe the client is connected to.

Return type str

vr_mode

`True` if Steam itself is running in VR mode.

Return type bool

Universe

class `steampak.libsteam.resources.utils.`**Universe**

BETA = 2

DEV = 4

INTERNAL = 3

INVALID = 0

MAX = 5

PUBLIC = 1

aliases = {0: 'invalid', 1: 'public', 2: 'beta', 3: 'internal', 4: 'dev', 5: 'max'}

get_alias (*item_id*)

Returns item alias by ID.

Parameters **item_id** (*int*) – Item ID.

Return type str

Notification Position

class steampak.libsteam.resources.utils.**NotificationPosition**

BOTTOM_LEFT = 2

BOTTOM_RIGHT = 3

TOP_LEFT = 0

TOP_RIGHT = 1

Index