GOG Galaxy Integrations API

Release 0.56

Galaxy team

Nov 07, 2019

CONTENTS

1 GOG Galaxy Integrations Python API **1**
- 1.1 Features . . . 1
- 1.2 Platform Id's . . . 1
- 1.3 Basic usage . . . 1
- 1.4 Deployment . . . 2
- 1.5 Legal Notice . . . 3

2 galaxy.api **5**
- 2.1 plugin . . . 5
- 2.2 types . . . 13
- 2.3 consts . . . 16
- 2.4 errors . . . 19

3 galaxy.http **21**

4 PLATFORM ID LIST **23**

5 Index **25**

Python Module Index **27**

Index **29**

CHAPTER

ONE

GOG GALAXY INTEGRATIONS PYTHON API

This Python library allows developers to easily build community integrations for various gaming platforms with GOG Galaxy 2.0.

1.1 Features

Each integration in GOG Galaxy 2.0 comes as a separate Python script and is launched as a separate process that needs to communicate with the main instance of GOG Galaxy 2.0.

The provided features are:

- multistep authorization using a browser built into GOG Galaxy 2.0
- support for GOG Galaxy 2.0 features:
  - importing owned and detecting installed games
  - installing and launching games
  - importing achievements and game time
  - importing friends lists and statuses
  - importing friends recommendations list
  - receiving and sending chat messages
- cache storage

1.2 Platform Id's

Each integration can implement only one platform. Each integration must declare which platform it's integrating.

PLATFORM ID LIST

1.3 Basic usage

Each integration should inherit from the *Plugin* class. Supported methods like *get_owned_games()* should be overwritten - they are called from the GOG Galaxy client at the appropriate times. Each of those methods can raise exceptions inherited from the *ApplicationError*. Communication between an integration and the client is also possible with the use of notifications, for example: *update_local_game_status()*.

```
import sys
from galaxy.api.plugin import Plugin, create_and_run_plugin
from galaxy.api.consts import Platform

class PluginExample(Plugin):
    def __init__(self, reader, writer, token):
        super().__init__(
            Platform.Generic, # Choose platform from available list
            "0.1", # Version
            reader,
            writer,
            token
        )

    # implement methods
    async def authenticate(self, stored_credentials=None):
        pass

def main():
    create_and_run_plugin(PluginExample, sys.argv)

# run plugin event loop
if __name__ == "__main__":
    main()
```

1.4 Deployment

The client has a built-in Python 3.7 interpreter, so integrations are delivered as Python modules. In order to be found by GOG Galaxy 2.0 an integration folder should be placed in *lookup directory*. Beside all the Python files, the integration folder must contain *manifest.json* and all third-party dependencies. See an *exemplary structure*.

1.4.1 Lookup directory

- Windows:

  ```
  %localappdata%\GOG.com\Galaxy\plugins\installed
  ```

- macOS:

  ```
  ~/Library/Application Support/GOG.com/Galaxy/plugins/installed
  ```

1.4.2 Manifest

Obligatory JSON file to be placed in an integration folder.

```
{
    "name": "Example plugin",
    "platform": "generic",
    "guid": "UNIQUE-GUID",
    "version": "0.1",
    "description": "Example plugin",
    "author": "Name",
    "email": "author@email.com",
```

(continues on next page)

(continued from previous page)

```
    "url": "https://github.com/user/galaxy-plugin-example",
    "script": "plugin.py"
}
```

property	description
guid	
description	
url	
script	path of the entry point module, relative to the integration folder

1.4.3 Dependencies

All third-party packages (packages not included in the Python 3.7 standard library) should be deployed along with plugin files. Use the following command structure:

```
pip install DEP --target DIR --implementation cp --python-version 37
```

For example, a plugin that uses *requests* could have the following structure:

```
installed
└── my_integration
    ├── galaxy
    │   └── api
    ├── requests
    │   └── ...
    ├── plugin.py
    └── manifest.json
```

1.5 Legal Notice

By integrating or attempting to integrate any applications or content with or into GOG Galaxy 2.0 you represent that such application or content is your original creation (other than any software made available by GOG) and/or that you have all necessary rights to grant such applicable rights to the relevant community integration to GOG and to GOG Galaxy 2.0 end users for the purpose of use of such community integration and that such community integration comply with any third party license and other requirements including compliance with applicable laws.

CHAPTER

TWO

GALAXY.API

2.1 plugin

class galaxy.api.plugin.**Plugin**(*platform*, *version*, *reader*, *writer*, *handshake_token*)

Use and override methods of this class to create a new platform integration.

persistent_cache

The cache is only available after the *handshake_complete()* is called.

Return type Dict[str, str]

close()

Return type None

create_task(*coro*, *description*)

Wrapper around asyncio.create_task - takes care of canceling tasks on shutdown

store_credentials(*credentials*)

Notify the client to store authentication credentials. Credentials are passed on the next authenticate call.

Parameters **credentials** (Dict[str, Any]) – credentials that client will store; they are stored locally on a user pc

Example use case of store_credentials:

```
async def pass_login_credentials(self, step, credentials, cookies):
    if self.got_everything(credentials,cookies):
        user_data = await self.parse_credentials(credentials,cookies)
    else:
        next_params = self.get_next_params(credentials,cookies)
        next_cookies = self.get_next_cookies(credentials,cookies)
        return NextStep("web_session", next_params, cookies=next_cookies)
    self.store_credentials(user_data['credentials'])
    return Authentication(user_data['userId'], user_data['username'])
```

Return type None

add_game(*game*)

Notify the client to add game to the list of owned games of the currently authenticated user.

Parameters **game** (*Game*) – Game to add to the list of owned games

Example use case of add_game:

```
async def check_for_new_games(self):
    games = await self.get_owned_games()
    for game in games:
        if game not in self.owned_games_cache:
            self.owned_games_cache.append(game)
            self.add_game(game)
```

Return type None

remove_game(*game_id*)

Notify the client to remove game from the list of owned games of the currently authenticated user.

Parameters **game_id** (str) – the id of the game to remove from the list of owned games

Example use case of remove_game:

```
async def check_for_removed_games(self):
    games = await self.get_owned_games()
    for game in self.owned_games_cache:
        if game not in games:
            self.owned_games_cache.remove(game)
            self.remove_game(game.game_id)
```

Return type None

update_game(*game*)

Notify the client to update the status of a game owned by the currently authenticated user.

Parameters **game** (*Game*) – Game to update

Return type None

unlock_achievement(*game_id*, *achievement*)

Notify the client to unlock an achievement for a specific game.

Parameters

- **game_id** (str) – the id of the game for which to unlock an achievement.
- **achievement** (*Achievement*) – achievement to unlock.

Return type None

update_local_game_status(*local_game*)

Notify the client to update the status of a local game.

Parameters **local_game** (*LocalGame*) – the LocalGame to update

Example use case triggered by the *tick()* method:

```
async def _check_statuses(self):
    for game in await self._get_local_games():
        if game.status == self._cached_game_statuses.get(game.id):
            continue
        self.update_local_game_status(LocalGame(game.id, game.status))
        self._cached_games_statuses[game.id] = game.status
    await asyncio.sleep(5)  # interval

def tick(self):
```

(continues on next page)

(continued from previous page)

```
    if self._check_statuses_task is None or self._check_statuses_task.done():
        self._check_statuses_task = asyncio.create_task(self._check_
↪statuses())
```

Return type None

add_friend(*user*)
Notify the client to add a user to friends list of the currently authenticated user.

Parameters **user** (*UserInfo*) – UserInfo of a user that the client will add to friends list

Return type None

remove_friend(*user_id*)
Notify the client to remove a user from friends list of the currently authenticated user.

Parameters **user_id** (str) – id of the user to remove from friends list

Return type None

update_game_time(*game_time*)
Notify the client to update game time for a game.

Parameters **game_time** (*GameTime*) – game time to update

Return type None

lost_authentication()
Notify the client that integration has lost authentication for the current user and is unable to perform actions which would require it.

Return type None

push_cache()
Push local copy of the persistent cache to the GOG Galaxy Client replacing existing one.

Return type None

handshake_complete()
This method is called right after the handshake with the GOG Galaxy Client is complete and before any other operations are called by the GOG Galaxy Client. Persistent cache is available when this method is called. Override it if you need to do additional plugin initializations. This method is called internally.

Return type None

tick()
This method is called periodically. Override it to implement periodical non-blocking tasks. This method is called internally.

Example of possible override of the method:

```
def tick(self):
    if not self.checking_for_new_games:
        asyncio.create_task(self.check_for_new_games())
    if not self.checking_for_removed_games:
        asyncio.create_task(self.check_for_removed_games())
    if not self.updating_game_statuses:
        asyncio.create_task(self.update_game_statuses())
```

Return type None

achievements_import_complete()
Override this method to handle operations after achievements import is finished (like updating cache).

game_times_import_complete()
Override this method to handle operations after game times import is finished (like updating cache).

Return type None

game_library_settings_import_complete()
Override this method to handle operations after game library settings import is finished (like updating cache).

Return type None

coroutine authenticate(*self*, *stored_credentials=None*)
Override this method to handle user authentication. This method should either return *Authentication* if the authentication is finished or *NextStep* if it requires going to another url. This method is called by the GOG Galaxy Client.

Parameters **stored_credentials** (Optional[Dict[~KT, ~VT]]) – If the client received any credentials to store locally in the previous session they will be passed here as a parameter.

Example of possible override of the method:

```
async def authenticate(self, stored_credentials=None):
    if not stored_credentials:
        return NextStep("web_session", PARAMS, cookies=COOKIES)
    else:
        try:
            user_data = self._authenticate(stored_credentials)
        except AccessDenied:
            raise InvalidCredentials()
    return Authentication(user_data['userId'], user_data['username'])
```

Return type Union[*NextStep*, *Authentication*]

coroutine get_friends(*self*)
Override this method to return the friends list of the currently authenticated user. This method is called by the GOG Galaxy Client.

Example of possible override of the method:

```
async def get_friends(self):
    if not self._http_client.is_authenticated():
        raise AuthenticationRequired()

    friends = self.retrieve_friends()
    return friends
```

Return type List[*UserInfo*]

coroutine get_game_library_settings(*self*, *game_id*, *context*)
Override this method to return the game library settings for the game identified by the provided game_id. This method is called by import task initialized by GOG Galaxy Client.

Parameters

- **game_id** (str) – the id of the game for which the game library settings are imported

- **context** (Any) – the value returned from *prepare_game_library_settings_context()*

Return type *GameLibrarySettings*

Returns GameLibrarySettings object

coroutine get_game_time(*self*, *game_id*, *context*)

Override this method to return the game time for the game identified by the provided game_id. This method is called by import task initialized by GOG Galaxy Client.

Parameters

- **game_id** (str) – the id of the game for which the game time is returned
- **context** (Any) – the value returned from *prepare_game_times_context()*

Return type *GameTime*

Returns GameTime object

coroutine get_local_games(*self*)

Override this method to return the list of games present locally on the users pc. This method is called by the GOG Galaxy Client.

Example of possible override of the method:

```
async def get_local_games(self):
    local_games = []
    for game in self.games_present_on_user_pc:
        local_game = LocalGame()
        local_game.game_id = game.id
        local_game.local_game_state = game.get_installation_status()
        local_games.append(local_game)
    return local_games
```

Return type List[*LocalGame*]

coroutine get_os_compatibility(*self*, *game_id*, *context*)

Override this method to return the OS compatibility for the game with the provided game_id. This method is called by import task initialized by GOG Galaxy Client.

Parameters

- **game_id** (str) – the id of the game for which the game os compatibility is imported
- **context** (Any) – the value returned from *prepare_os_compatibility_context()*

Return type Optional[*OSCompatibility*]

Returns OSCompatibility flags indicating compatible OSs, or None if compatibility is not know

coroutine get_owned_games(*self*)

Override this method to return owned games for currently logged in user. This method is called by the GOG Galaxy Client.

Example of possible override of the method:

```
async def get_owned_games(self):
    if not self.authenticated():
        raise AuthenticationRequired()

    games = self.retrieve_owned_games()
    return games
```

Return type List[*Game*]

coroutine get_user_presence(*self*, *user_id*, *context*)
Override this method to return presence information for the user with the provided user_id. This method is called by import task initialized by GOG Galaxy Client.

Parameters

- **user_id** (str) – the id of the user for whom presence information is imported
- **context** (Any) – the value returned from *prepare_user_presence_context()*

Return type *UserPresence*

Returns UserPresence presence information of the provided user

coroutine install_game(*self*, *game_id*)
Override this method to install the game identified by the provided game_id. This method is called by the GOG Galaxy Client.

Parameters **game_id** (*str*) – the id of the game to install

Example of possible override of the method:

```
async def install_game(self, game_id):
    await self.open_uri(f"start client://installgame/{game_id}")
```

Return type None

coroutine launch_game(*self*, *game_id*)
Override this method to launch the game identified by the provided game_id. This method is called by the GOG Galaxy Client.

Parameters **game_id** (*str*) – the id of the game to launch

Example of possible override of the method:

```
async def launch_game(self, game_id):
    await self.open_uri(f"start client://launchgame/{game_id}")
```

Return type None

coroutine launch_platform_client(*self*)
Override this method to launch platform client. Preferably minimized to tray. This method is called by the GOG Galaxy Client.

Return type None

coroutine pass_login_credentials(*self*, *step*, *credentials*, *cookies*)
This method is called if we return galaxy.api.types.NextStep from authenticate or from pass_login_credentials. This method's parameters provide the data extracted from the web page navigation that previous NextStep finished on. This method should either return galaxy.api.types.Authentication if the authentication is finished or galaxy.api.types.NextStep if it requires going to another cef url. This method is called by the GOG Galaxy Client.

Parameters

- **step** (str) – deprecated.
- **credentials** (Dict[str, str]) – end_uri previous NextStep finished on.
- **cookies** (List[Dict[str, str]]) – cookies extracted from the end_uri site.

Example of possible override of the method:

```
async def pass_login_credentials(self, step, credentials, cookies):
    if self.got_everything(credentials,cookies):
        user_data = await self.parse_credentials(credentials,cookies)
    else:
        next_params = self.get_next_params(credentials,cookies)
        next_cookies = self.get_next_cookies(credentials,cookies)
        return NextStep("web_session", next_params, cookies=next_cookies)
    self.store_credentials(user_data['credentials'])
    return Authentication(user_data['userId'], user_data['username'])
```

Return type Union[*NextStep*, *Authentication*]

coroutine prepare_achievements_context(*self*, *game_ids*)

Override this method to prepare context for get_unlocked_achievements. This allows for optimizations like batch requests to platform API. Default implementation returns None.

Parameters **game_ids** (List[str]) – the ids of the games for which achievements are imported

Return type Any

Returns context

coroutine prepare_game_library_settings_context(*self*, *game_ids*)

Override this method to prepare context for get_game_library_settings. This allows for optimizations like batch requests to platform API. Default implementation returns None.

Parameters **game_ids** (List[str]) – the ids of the games for which game library settings are imported

Return type Any

Returns context

coroutine prepare_game_times_context(*self*, *game_ids*)

Override this method to prepare context for get_game_time. This allows for optimizations like batch requests to platform API. Default implementation returns None.

Parameters **game_ids** (List[str]) – the ids of the games for which game time are imported

Return type Any

Returns context

coroutine prepare_os_compatibility_context(*self*, *game_ids*)

Override this method to prepare context for get_os_compatibility. This allows for optimizations like batch requests to platform API. Default implementation returns None.

Parameters **game_ids** (List[str]) – the ids of the games for which game os compatibility is imported

Return type Any

Returns context

coroutine prepare_user_presence_context(*self*, *user_ids*)

Override this method to prepare context for get_user_presence. This allows for optimizations like batch requests to platform API. Default implementation returns None.

Parameters **user_ids** (List[str]) – the ids of the users for whom presence information is imported

Return type Any

Returns context

coroutine refresh_credentials(*self*, *params*, *sensitive_params*)

Return type Dict[str, Any]

coroutine run()
Plugin's main coroutine.

coroutine shutdown(*self*)
This method is called on integration shutdown. Override it to implement tear down. This method is called by the GOG Galaxy Client.

Return type None

coroutine shutdown_platform_client(*self*)
Override this method to gracefully terminate platform client. This method is called by the GOG Galaxy Client.

Return type None

coroutine uninstall_game(*self*, *game_id*)
Override this method to uninstall the game identified by the provided game_id. This method is called by the GOG Galaxy Client.

Parameters **game_id** (*str*) – the id of the game to uninstall

Example of possible override of the method:

```
async def uninstall_game(self, game_id):
    await self.open_uri(f"start client://uninstallgame/{game_id}")
```

Return type None

coroutine wait_closed(*self*)

Return type None

os_compatibility_import_complete()
Override this method to handle operations after OS compatibility import is finished (like updating cache).

Return type None

user_presence_import_complete()
Override this method to handle operations after presence import is finished (like updating cache).

Return type None

galaxy.api.plugin.**create_and_run_plugin**(*plugin_class*, *argv*)
Call this method as an entry point for the implemented integration.

Parameters

- **plugin_class** – your plugin class.
- **argv** – command line arguments with which the script was started.

Example of possible use of the method:

```
def main():
    create_and_run_plugin(PlatformPlugin, sys.argv)

if __name__ == "__main__":
    main()
```

2.2 types

class galaxy.api.types.**Authentication**(*user_id*, *user_name*)

Return this from *authenticate()* or *pass_login_credentials()* to inform the client that authentication has successfully finished.

Parameters

- **user_id** (str) – id of the authenticated user
- **user_name** (str) – username of the authenticated user

class galaxy.api.types.**Cookie**(*name*, *value*, *domain=None*, *path=None*)

Parameters

- **name** (str) – name of the cookie
- **value** (str) – value of the cookie
- **domain** (Optional[str]) – optional domain of the cookie
- **path** (Optional[str]) – optional path of the cookie

domain = None

path = None

class galaxy.api.types.**NextStep**(*next_step*, *auth_params*, *cookies=None*, *js=None*)

Return this from *authenticate()* or *pass_login_credentials()* to open client built-in browser with given url. For example:

```
PARAMS = {
    "window_title": "Login to platform",
    "window_width": 800,
    "window_height": 600,
    "start_uri": URL,
    "end_uri_regex": r"^https://platform_website\.com/.*"
}

JS = {r"^https://platform_website\.com/.*": [
    r'''
        location.reload();
    '''
]}

COOKIES = [Cookie("Cookie1", "ok", ".platform.com"),
    Cookie("Cookie2", "ok", ".platform.com")
    ]

async def authenticate(self, stored_credentials=None):
    if not stored_credentials:
        return NextStep("web_session", PARAMS, cookies=COOKIES, js=JS)
```

Parameters

- **auth_params** (Dict[str, str]) – configuration options: {"window_title": str, "window_width": str, "window_height": int, "start_uri": int, "end_uri_regex": str}
- **cookies** (Optional[List[*Cookie*]]) – browser initial set of cookies
- **js** (Optional[Dict[str, List[str]]]) – a map of the url regex patterns into the list of *js* scripts that should be executed on every document at given step of internal browser authentication.

cookies = None

js = None

class galaxy.api.types.**LicenseInfo**(*license_type*, *owner=None*)
Information about the license of related product.

Parameters

- **license_type** (*LicenseType*) – type of license
- **owner** (Optional[str]) – optional owner of the related product, defaults to currently authenticated user

owner = None

class galaxy.api.types.**Dlc**(*dlc_id*, *dlc_title*, *license_info*)
Downloadable content object.

Parameters

- **dlc_id** (str) – id of the dlc
- **dlc_title** (str) – title of the dlc
- **license_info** (*LicenseInfo*) – information about the license attached to the dlc

class galaxy.api.types.**Game**(*game_id*, *game_title*, *dlcs*, *license_info*)
Game object.

Parameters

- **game_id** (str) – unique identifier of the game, this will be passed as parameter for methods such as launch_game
- **game_title** (str) – title of the game
- **dlcs** (Optional[List[*Dlc*]]) – list of dlcs available for the game
- **license_info** (*LicenseInfo*) – information about the license attached to the game

class galaxy.api.types.**Achievement**(*unlock_time*, *achievement_id=None*, *achievement_name=None*)
Achievement, has to be initialized with either id or name.

Parameters

- **unlock_time** (int) – unlock time of the achievement
- **achievement_id** (Optional[str]) – optional id of the achievement
- **achievement_name** (Optional[str]) – optional name of the achievement

achievement_id = None

achievement_name = None

class galaxy.api.types.**LocalGame**(*game_id*, *local_game_state*)
Game locally present on the authenticated user's computer.

Parameters

- **game_id** (str) – id of the game
- **local_game_state** (*LocalGameState*) – state of the game

class galaxy.api.types.**FriendInfo**(*user_id*, *user_name*)
Deprecated since version 0.56.

Use: *UserInfo*. Information about a friend of the currently authenticated user.

Parameters

- **user_id** (str) – id of the user
- **user_name** (str) – username of the user

class galaxy.api.types.**UserInfo**(*user_id*, *user_name*)
Information about a user of related user.

Parameters

- **user_id** (str) – id of the user
- **user_name** (str) – username of the user

class galaxy.api.types.**GameTime**(*game_id*, *time_played*, *last_played_time*)
Game time of a game, defines the total time spent in the game and the last time the game was played.

Parameters

- **game_id** (str) – id of the related game
- **time_played** (Optional[int]) – the total time spent in the game in **minutes**
- **last_time_played** – last time the game was played (**unix timestamp**)

class galaxy.api.types.**GameLibrarySettings**(*game_id*, *tags*, *hidden*)
Library settings of a game, defines assigned tags and visibility flag.

Parameters

- **game_id** (str) – id of the related game
- **tags** (Optional[List[str]]) – collection of tags assigned to the game
- **hidden** (Optional[bool]) – indicates if the game should be hidden in GOG Galaxy application

class galaxy.api.types.**UserPresence**(*presence_state*, *game_id=None*, *game_title=None*, *presence_status=None*)
Presence information of a user.

Parameters

- **presence_state** (*PresenceState*) – the state of the user
- **game_id** (Optional[str]) – id of the game a user is currently in
- **game_title** (Optional[str]) – name of the game a user is currently in
- **presence_status** (Optional[str]) – detailed user's presence description

game_id = None

game_title = None

```
presence_status = None
```

2.3 consts

class galaxy.api.consts.**Platform**

Bases: enum.Enum

Supported gaming platforms

```
Unknown = 'unknown'
Gog = 'gog'
Steam = 'steam'
Psn = 'psn'
XBoxOne = 'xboxone'
Generic = 'generic'
Origin = 'origin'
Uplay = 'uplay'
Battlenet = 'battlenet'
Epic = 'epic'
Bethesda = 'bethesda'
ParadoxPlaza = 'paradox'
HumbleBundle = 'humble'
Kartridge = 'kartridge'
ItchIo = 'itch'
NintendoSwitch = 'nswitch'
NintendoWiiU = 'nwiiu'
NintendoWii = 'nwii'
NintendoGameCube = 'ncube'
RiotGames = 'riot'
Wargaming = 'wargaming'
NintendoGameBoy = 'ngameboy'
Atari = 'atari'
Amiga = 'amiga'
SuperNintendoEntertainmentSystem = 'snes'
Beamdog = 'beamdog'
Direct2Drive = 'd2d'
Discord = 'discord'
DotEmu = 'dotemu'
GameHouse = 'gamehouse'
```

```
    GreenManGaming = 'gmg'
    WePlay = 'weplay'
    ZxSpectrum = 'zx'
    ColecoVision = 'vision'
    NintendoEntertainmentSystem = 'nes'
    SegaMasterSystem = 'sms'
    Commodore64 = 'c64'
    PcEngine = 'pce'
    SegaGenesis = 'segag'
    NeoGeo = 'neo'
    Sega32X = 'sega32'
    SegaCd = 'segacd'
    SegaSaturn = 'saturn'
    PlayStation = 'psx'
    PlayStation2 = 'ps2'
    Nintendo64 = 'n64'
    AtariJaguar = 'jaguar'
    SegaDreamcast = 'dc'
    Xbox = 'xboxog'
    Amazon = 'amazon'
    GamersGate = 'gg'
    Newegg = 'egg'
    BestBuy = 'bb'
    GameUk = 'gameuk'
    Fanatical = 'fanatical'
    PlayAsia = 'playasia'
    Stadia = 'stadia'
    Arc = 'arc'
    ElderScrollsOnline = 'eso'
    Glyph = 'glyph'
    AionLegionsOfWar = 'aionl'
    Aion = 'aion'
    BladeAndSoul = 'blade'
    GuildWars = 'gw'
    GuildWars2 = 'gw2'
    Lineage2 = 'lin2'
```

FinalFantasy11 = 'ffxi'

FinalFantasy14 = 'ffxiv'

TotalWar = 'totalwar'

WindowsStore = 'winstore'

EliteDangerous = 'elites'

StarCitizen = 'star'

PlayStationPortable = 'psp'

PlayStationVita = 'psvita'

NintendoDs = 'nds'

Nintendo3Ds = '3ds'

PathOfExile = 'pathofexile'

Twitch = 'twitch'

Minecraft = 'minecraft'

GameSessions = 'gamesessions'

Nuuvem = 'nuuvem'

FXStore = 'fxstore'

IndieGala = 'indiegala'

Playfire = 'playfire'

Oculus = 'oculus'

Test = 'test'

Rockstar = 'rockstar'

class galaxy.api.consts.**LicenseType**

Bases: enum.Enum

Possible game license types, understandable for the GOG Galaxy client.

Unknown = 'Unknown'

SinglePurchase = 'SinglePurchase'

FreeToPlay = 'FreeToPlay'

OtherUserLicense = 'OtherUserLicense'

class galaxy.api.consts.**LocalGameState**

Bases: enum.Flag

Possible states that a local game can be in. For example a game which is both installed and currently running should have its state set as a "bitwise or" of Running and Installed flags: local_game_state=<LocalGameState.Running|Installed: 3>

None_ = 0

Installed = 1

Running = 2

class galaxy.api.consts.**OSCompatibility**
Bases: enum.Flag

Possible game OS compatibility. Use "bitwise or" to express multiple OSs compatibility, e.g. `os=OSCompatibility.Windows|OSCompatibility.MacOS`

Windows = 1

MacOS = 2

Linux = 4

class galaxy.api.consts.**PresenceState**
Bases: enum.Enum

"Possible states of a user.

Unknown = 'unknown'

Online = 'online'

Offline = 'offline'

Away = 'away'

2.4 errors

exception galaxy.api.jsonrpc.**ApplicationError**(*code*, *message*, *data*)
Bases: galaxy.api.jsonrpc.JsonRpcError

exception galaxy.api.jsonrpc.**UnknownError**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**AuthenticationRequired**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**BackendNotAvailable**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**BackendTimeout**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**BackendError**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**UnknownBackendResponse**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**TooManyRequests**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**InvalidCredentials**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**NetworkError**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**LoggedInElsewhere**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**ProtocolError**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**TemporaryBlocked**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**Banned**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**AccessDenied**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**FailedParsingManifest**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**TooManyMessagesSent**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**IncoherentLastMessage**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**MessageNotFound**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

exception galaxy.api.errors.**ImportInProgress**(*data=None*)
Bases: *galaxy.api.jsonrpc.ApplicationError*

CHAPTER

THREE

GALAXY.HTTP

This module standarize http traffic and the error handling for further communication with the GOG Galaxy 2.0.

It is recommended to use provided convenient methods for HTTP requests, especially when dealing with authorized sessions. Examplary simple web service could looks like:

```
import logging
from galaxy.http import create_client_session, handle_exception

class BackendClient:
    AUTH_URL = 'my-integration.com/auth'
    HEADERS = {
        "My-Custom-Header": "true",
    }
    def __init__(self):
        self._session = create_client_session(headers=self.HEADERS)

    async def authenticate(self):
        await self._session.request('POST', self.AUTH_URL)

    async def close(self):
        # to be called on plugin shutdown
        await self._session.close()

    async def _authorized_request(self, method, url, *args, **kwargs):
        with handle_exceptions():
            return await self._session.request(method, url, *args, **kwargs)
```

galaxy.http.**DEFAULT_LIMIT = 20**
Default limit of the simultaneous connections for ssl connector.

galaxy.http.**DEFAULT_TIMEOUT = 60**
Default timeout in seconds used for client session.

class galaxy.http.**HttpClient**(*limit=20*, *timeout=aiohttp.ClientTimeout*, *cookie_jar=None*)
Bases: object

Deprecated since version 0.41: Use http module functions instead

__init__(*limit=20*, *timeout=aiohttp.ClientTimeout*, *cookie_jar=None*)

coroutine close()
Closes connection. Should be called in *shutdown()*

coroutine request(*method*, *url*, **args*, ***kwargs*)

galaxy.http.**create_tcp_connector**(**args*, ***kwargs*)

Creates TCP connector with resonable defaults. For details about available parameters refer to aiohttp.TCPConnector

Return type aiohttp.TCPConnector

galaxy.http.**create_client_session**(**args*, ***kwargs*)

Creates client session with resonable defaults. For details about available parameters refer to aiohttp.ClientSession

Examplary customization:

```
from galaxy.http import create_client_session, create_tcp_connector

session = create_client_session(
    headers={
        "Keep-Alive": "true"
    },
    connector=create_tcp_connector(limit=40),
    timeout=100)
```

Return type aiohttp.ClientSession

galaxy.http.**handle_exception**()

Context manager translating network related exceptions to custom *errors*.

CHAPTER

FOUR

PLATFORM ID LIST

Platform ID list for GOG Galaxy 2.0 Integrations

ID	Name
steam	Steam
psn	PlayStation Network
xboxone	Xbox Live
generic	Manually added games
origin	Origin
uplay	Uplay
battlenet	Battle.net
epic	Epic Games Store
bethesda	Bethesda.net
paradox	Paradox Plaza
humble	Humble Bundle
kartridge	Kartridge
itch	Itch.io
nswitch	Nintendo Switch
nwiiu	Nintendo Wii U
nwii	Nintendo Wii
ncube	Nintendo GameCube
riot	Riot
wargaming	Wargaming
ngameboy	Nintendo Game Boy
atari	Atari
amiga	Amiga
snes	SNES
beamdog	Beamdog
d2d	Direct2Drive
discord	Discord
dotemu	DotEmu
gamehouse	GameHouse
gmg	Green Man Gaming
weplay	WePlay
zx	Zx Spectrum PC
vision	ColecoVision
nes	NES
sms	Sega Master System
c64	Commodore 64

Continued on next page

Table 1 – continued from previous page

ID	Name
pce	PC Engine
segag	Sega Genesis
neo	NeoGeo
sega32	Sega 32X
segacd	Sega CD
3do	3DO Interactive
saturn	SegaSaturn
psx	Sony PlayStation
ps2	Sony PlayStation 2
n64	Nintendo64
jaguar	Atari Jaguar
dc	Sega Dreamcast
xboxog	Original Xbox games
amazon	Amazon
gg	GamersGate
egg	Newegg
bb	BestBuy
gameuk	Game UK
fanatical	Fanatical store
playasia	Play-Asia
stadia	Google Stadia
arc	ARC
eso	ESO
glyph	Trion World
aionl	Aion: Legions of War
aion	Aion
blade	Blade & Soul
gw	Guild Wars
gw2	Guild Wars 2
lin2	Lineage 2
ffxi	Final Fantasy XI
ffxiv	Final Fantasy XIV
totalwar	Total War
winstore	Windows Store
elites	Elite Dangerous
star	Star Citizen
psp	PlayStation Portable
psvita	PlayStation Vita
nds	Nintendo DS
3ds	Nintendo 3DS
pathofexile	Path of Exile

CHAPTER

FIVE

INDEX

- genindex

PYTHON MODULE INDEX

g

galaxy.api.consts, 16
galaxy.api.errors, 19
galaxy.api.plugin, 5
galaxy.api.types, 13
galaxy.http, 21

INDEX

Symbols

__init__() (*galaxy.http.HttpClient method*), 21

A

AccessDenied, 20
Achievement (*class in galaxy.api.types*), 14
achievement_id (*galaxy.api.types.Achievement attribute*), 14
achievement_name (*galaxy.api.types.Achievement attribute*), 14
achievements_import_complete() (*galaxy.api.plugin.Plugin method*), 7
add_friend() (*galaxy.api.plugin.Plugin method*), 7
add_game() (*galaxy.api.plugin.Plugin method*), 5
Aion (*galaxy.api.consts.Platform attribute*), 17
AionLegionsOfWar (*galaxy.api.consts.Platform attribute*), 17
Amazon (*galaxy.api.consts.Platform attribute*), 17
Amiga (*galaxy.api.consts.Platform attribute*), 16
ApplicationError, 19
Arc (*galaxy.api.consts.Platform attribute*), 17
Atari (*galaxy.api.consts.Platform attribute*), 16
AtariJaguar (*galaxy.api.consts.Platform attribute*), 17
authenticate() (*galaxy.api.plugin.Plugin method*), 8
Authentication (*class in galaxy.api.types*), 13
AuthenticationRequired, 19
Away (*galaxy.api.consts.PresenceState attribute*), 19

B

BackendError, 19
BackendNotAvailable, 19
BackendTimeout, 19
Banned, 20
Battlenet (*galaxy.api.consts.Platform attribute*), 16
Beamdog (*galaxy.api.consts.Platform attribute*), 16
BestBuy (*galaxy.api.consts.Platform attribute*), 17
Bethesda (*galaxy.api.consts.Platform attribute*), 16
BladeAndSoul (*galaxy.api.consts.Platform attribute*), 17

C

close() (*galaxy.api.plugin.Plugin method*), 5
close() (*galaxy.http.HttpClient method*), 21
ColecoVision (*galaxy.api.consts.Platform attribute*), 17
Commodore64 (*galaxy.api.consts.Platform attribute*), 17
Cookie (*class in galaxy.api.types*), 13
cookies (*galaxy.api.types.NextStep attribute*), 14
create_and_run_plugin() (*in module galaxy.api.plugin*), 12
create_client_session() (*in module galaxy.http*), 22
create_task() (*galaxy.api.plugin.Plugin method*), 5
create_tcp_connector() (*in module galaxy.http*), 21

D

DEFAULT_LIMIT (*in module galaxy.http*), 21
DEFAULT_TIMEOUT (*in module galaxy.http*), 21
Direct2Drive (*galaxy.api.consts.Platform attribute*), 16
Discord (*galaxy.api.consts.Platform attribute*), 16
Dlc (*class in galaxy.api.types*), 14
domain (*galaxy.api.types.Cookie attribute*), 13
DotEmu (*galaxy.api.consts.Platform attribute*), 16

E

ElderScrollsOnline (*galaxy.api.consts.Platform attribute*), 17
EliteDangerous (*galaxy.api.consts.Platform attribute*), 18
Epic (*galaxy.api.consts.Platform attribute*), 16

F

FailedParsingManifest, 20
Fanatical (*galaxy.api.consts.Platform attribute*), 17
FinalFantasy11 (*galaxy.api.consts.Platform attribute*), 17
FinalFantasy14 (*galaxy.api.consts.Platform attribute*), 18

FreeToPlay (*galaxy.api.consts.LicenseType attribute*), 18
FriendInfo (*class in galaxy.api.types*), 15
FXStore (*galaxy.api.consts.Platform attribute*), 18

G

galaxy.api.consts (*module*), 16
galaxy.api.errors (*module*), 19
galaxy.api.plugin (*module*), 5
galaxy.api.types (*module*), 13
galaxy.http (*module*), 21
Game (*class in galaxy.api.types*), 14
game_id (*galaxy.api.types.UserPresence attribute*), 15
game_library_settings_import_complete() (*galaxy.api.plugin.Plugin method*), 8
game_times_import_complete() (*galaxy.api.plugin.Plugin method*), 8
game_title (*galaxy.api.types.UserPresence attribute*), 15
GameHouse (*galaxy.api.consts.Platform attribute*), 16
GameLibrarySettings (*class in galaxy.api.types*), 15
GamersGate (*galaxy.api.consts.Platform attribute*), 17
GameSessions (*galaxy.api.consts.Platform attribute*), 18
GameTime (*class in galaxy.api.types*), 15
GameUk (*galaxy.api.consts.Platform attribute*), 17
Generic (*galaxy.api.consts.Platform attribute*), 16
get_friends() (*galaxy.api.plugin.Plugin method*), 8
get_game_library_settings() (*galaxy.api.plugin.Plugin method*), 8
get_game_time() (*galaxy.api.plugin.Plugin method*), 9
get_local_games() (*galaxy.api.plugin.Plugin method*), 9
get_os_compatibility() (*galaxy.api.plugin.Plugin method*), 9
get_owned_games() (*galaxy.api.plugin.Plugin method*), 9
get_user_presence() (*galaxy.api.plugin.Plugin method*), 10
Glyph (*galaxy.api.consts.Platform attribute*), 17
Gog (*galaxy.api.consts.Platform attribute*), 16
GreenManGaming (*galaxy.api.consts.Platform attribute*), 16
GuildWars (*galaxy.api.consts.Platform attribute*), 17
GuildWars2 (*galaxy.api.consts.Platform attribute*), 17

H

handle_exception() (*in module galaxy.http*), 22
handshake_complete() (*galaxy.api.plugin.Plugin method*), 7
HttpClient (*class in galaxy.http*), 21
HumbleBundle (*galaxy.api.consts.Platform attribute*), 16

I

ImportInProgress, 20
IncoherentLastMessage, 20
IndieGala (*galaxy.api.consts.Platform attribute*), 18
install_game() (*galaxy.api.plugin.Plugin method*), 10
Installed (*galaxy.api.consts.LocalGameState attribute*), 18
InvalidCredentials, 19
ItchIo (*galaxy.api.consts.Platform attribute*), 16

J

js (*galaxy.api.types.NextStep attribute*), 14

K

Kartridge (*galaxy.api.consts.Platform attribute*), 16

L

launch_game() (*galaxy.api.plugin.Plugin method*), 10
launch_platform_client() (*galaxy.api.plugin.Plugin method*), 10
LicenseInfo (*class in galaxy.api.types*), 14
LicenseType (*class in galaxy.api.consts*), 18
Lineage2 (*galaxy.api.consts.Platform attribute*), 17
Linux (*galaxy.api.consts.OSCompatibility attribute*), 19
LocalGame (*class in galaxy.api.types*), 14
LocalGameState (*class in galaxy.api.consts*), 18
LoggedInElsewhere, 19
lost_authentication() (*galaxy.api.plugin.Plugin method*), 7

M

MacOS (*galaxy.api.consts.OSCompatibility attribute*), 19
MessageNotFound, 20
Minecraft (*galaxy.api.consts.Platform attribute*), 18

N

NeoGeo (*galaxy.api.consts.Platform attribute*), 17
NetworkError, 19
Newegg (*galaxy.api.consts.Platform attribute*), 17
NextStep (*class in galaxy.api.types*), 13
Nintendo3Ds (*galaxy.api.consts.Platform attribute*), 18
Nintendo64 (*galaxy.api.consts.Platform attribute*), 17
NintendoDs (*galaxy.api.consts.Platform attribute*), 18
NintendoEntertainmentSystem (*galaxy.api.consts.Platform attribute*), 17
NintendoGameBoy (*galaxy.api.consts.Platform attribute*), 16

NintendoGameCube (*galaxy.api.consts.Platform attribute*), 16
NintendoSwitch (*galaxy.api.consts.Platform attribute*), 16
NintendoWii (*galaxy.api.consts.Platform attribute*), 16
NintendoWiiU (*galaxy.api.consts.Platform attribute*), 16
None_ (*galaxy.api.consts.LocalGameState attribute*), 18
Nuuvem (*galaxy.api.consts.Platform attribute*), 18

O

Oculus (*galaxy.api.consts.Platform attribute*), 18
Offline (*galaxy.api.consts.PresenceState attribute*), 19
Online (*galaxy.api.consts.PresenceState attribute*), 19
Origin (*galaxy.api.consts.Platform attribute*), 16
os_compatibility_import_complete() (*galaxy.api.plugin.Plugin method*), 12
OSCompatibility (*class in galaxy.api.consts*), 18
OtherUserLicense (*galaxy.api.consts.LicenseType attribute*), 18
owner (*galaxy.api.types.LicenseInfo attribute*), 14

P

ParadoxPlaza (*galaxy.api.consts.Platform attribute*), 16
pass_login_credentials() (*galaxy.api.plugin.Plugin method*), 10
path (*galaxy.api.types.Cookie attribute*), 13
PathOfExile (*galaxy.api.consts.Platform attribute*), 18
PcEngine (*galaxy.api.consts.Platform attribute*), 17
persistent_cache (*galaxy.api.plugin.Plugin attribute*), 5
Platform (*class in galaxy.api.consts*), 16
PlayAsia (*galaxy.api.consts.Platform attribute*), 17
Playfire (*galaxy.api.consts.Platform attribute*), 18
PlayStation (*galaxy.api.consts.Platform attribute*), 17
PlayStation2 (*galaxy.api.consts.Platform attribute*), 17
PlayStationPortable (*galaxy.api.consts.Platform attribute*), 18
PlayStationVita (*galaxy.api.consts.Platform attribute*), 18
Plugin (*class in galaxy.api.plugin*), 5
prepare_achievements_context() (*galaxy.api.plugin.Plugin method*), 11
prepare_game_library_settings_context() (*galaxy.api.plugin.Plugin method*), 11
prepare_game_times_context() (*galaxy.api.plugin.Plugin method*), 11
prepare_os_compatibility_context() (*galaxy.api.plugin.Plugin method*), 11
prepare_user_presence_context() (*galaxy.api.plugin.Plugin method*), 11
presence_status (*galaxy.api.types.UserPresence attribute*), 15
PresenceState (*class in galaxy.api.consts*), 19
ProtocolError, 19
Psn (*galaxy.api.consts.Platform attribute*), 16
push_cache() (*galaxy.api.plugin.Plugin method*), 7

R

refresh_credentials() (*galaxy.api.plugin.Plugin method*), 12
remove_friend() (*galaxy.api.plugin.Plugin method*), 7
remove_game() (*galaxy.api.plugin.Plugin method*), 6
request() (*galaxy.http.HttpClient method*), 21
RiotGames (*galaxy.api.consts.Platform attribute*), 16
Rockstar (*galaxy.api.consts.Platform attribute*), 18
run() (*galaxy.api.plugin.Plugin method*), 12
Running (*galaxy.api.consts.LocalGameState attribute*), 18

S

Sega32X (*galaxy.api.consts.Platform attribute*), 17
SegaCd (*galaxy.api.consts.Platform attribute*), 17
SegaDreamcast (*galaxy.api.consts.Platform attribute*), 17
SegaGenesis (*galaxy.api.consts.Platform attribute*), 17
SegaMasterSystem (*galaxy.api.consts.Platform attribute*), 17
SegaSaturn (*galaxy.api.consts.Platform attribute*), 17
shutdown() (*galaxy.api.plugin.Plugin method*), 12
shutdown_platform_client() (*galaxy.api.plugin.Plugin method*), 12
SinglePurchase (*galaxy.api.consts.LicenseType attribute*), 18
Stadia (*galaxy.api.consts.Platform attribute*), 17
StarCitizen (*galaxy.api.consts.Platform attribute*), 18
Steam (*galaxy.api.consts.Platform attribute*), 16
store_credentials() (*galaxy.api.plugin.Plugin method*), 5
SuperNintendoEntertainmentSystem (*galaxy.api.consts.Platform attribute*), 16

T

TemporaryBlocked, 19
Test (*galaxy.api.consts.Platform attribute*), 18
tick() (*galaxy.api.plugin.Plugin method*), 7
TooManyMessagesSent, 20
TooManyRequests, 19
TotalWar (*galaxy.api.consts.Platform attribute*), 18
Twitch (*galaxy.api.consts.Platform attribute*), 18

U

uninstall_game() (*galaxy.api.plugin.Plugin method*), 12
Unknown (*galaxy.api.consts.LicenseType attribute*), 18
Unknown (*galaxy.api.consts.Platform attribute*), 16
Unknown (*galaxy.api.consts.PresenceState attribute*), 19
UnknownBackendResponse, 19
UnknownError, 19
unlock_achievement() (*galaxy.api.plugin.Plugin method*), 6
update_game() (*galaxy.api.plugin.Plugin method*), 6
update_game_time() (*galaxy.api.plugin.Plugin method*), 7
update_local_game_status() (*galaxy.api.plugin.Plugin method*), 6
Uplay (*galaxy.api.consts.Platform attribute*), 16
user_presence_import_complete() (*galaxy.api.plugin.Plugin method*), 12
UserInfo (*class in galaxy.api.types*), 15
UserPresence (*class in galaxy.api.types*), 15

W

wait_closed() (*galaxy.api.plugin.Plugin method*), 12
Wargaming (*galaxy.api.consts.Platform attribute*), 16
WePlay (*galaxy.api.consts.Platform attribute*), 17
Windows (*galaxy.api.consts.OSCompatibility attribute*), 19
WindowsStore (*galaxy.api.consts.Platform attribute*), 18

X

Xbox (*galaxy.api.consts.Platform attribute*), 17
XBoxOne (*galaxy.api.consts.Platform attribute*), 16

Z

ZxSpectrum (*galaxy.api.consts.Platform attribute*), 17