LinkedIn API for Python

Become a sponsor

Programmatically send messages, get jobs, and search profiles with a regular LinkedIn user account.

No “official” API access required - just use a valid LinkedIn account!

Caution: This library is not officially supported by LinkedIn. Using it might violate LinkedIn’s Terms of Service. Use it at your own risk.

Usage

API Documentation

latest

Key Concepts

Contents

URN ID vs public ID

While using the project, you'll come across two different types of identifier: URN IDs and public IDs.

URN ID

A URN ID is generally a number or something not human-readable. They will end up as part of a URN for a given entity. Here is an example of a LinkedIn URN for a profile:

```
urn:li:fs_miniProfile:ACoAABQ11fIBQLGQbB1V1XPBZJsRwfK5r1U2Rzt
```

In this case, "ACoAABQ11fIBQLGQbB1V1XPBZJsRwfK5r1U2Rzt" is the URN ID. When asked for a URN ID, don't provide a full URN. Instead, only provide the last item in the URN, delimited by ':' character (the URN ID).

Public ID

A public ID is generally a string and something that is human-readable. For example, for the LinkedIn profile at the URL https://www.linkedin.com/in/tom-quirk, the "public ID" is "tom-quirk". The same applies for other entities across LinkedIn

Examples

Contents

Get a profile and their connections, then send a message

```python
import json

from linkedin_api import Linkedin

with open("credentials.json", "r") as f:
    credentials = json.load(f)

if credentials:
    linkedin = Linkedin(credentials["username"], credentials["password"])

    profile = linkedin.get_profile("ACoAABQ11fIBQLGQbB1V1XPBZJsRwfK5r1U2Rzw")
    profile["contact_info"] = linkedin.get_profile_contact_info(
        "ACoAABQ11fIBQLGQbB1V1XPBZJsRwfK5r1U2Rzw"
    )
    connections = linkedin.get_profile_connections(profile["profile_id"])
    # send a message
    linkedin.send_message(
        recipients=[profile["profile_id"]],
        message="Hello, Hola, Namaste, Hii, Bonjour, Guten Tag",
    )
```

API Docs

Contents

linkedin-api

class linkedin_api.**Linkedin**(*username: str*, *password: str*, ***, *authenticate=True*, *refresh_cookies=False*, *debug=False*, *proxies={}*, *cookies=None*, *cookies_dir: str = ''*)

Class for accessing the LinkedIn API.

Parameters:

- **username** (*str*) – Username of LinkedIn account.
- **password** (*str*) – Password of LinkedIn account.

add_connection(*profile_public_id: str*, *message=''*, *profile_urn=None*)

Add a given profile id as a connection.

Parameters:

- **profile_public_id** (*str*) – public ID of a LinkedIn profile
- **message** – message to send along with connection request
- **profile_urn** (*str, optional*) – member URN for the given LinkedIn profile

Returns:

Error state. True if error occurred

Return type:

boolean

follow_company(*following_state_urn*, *following=True*)

Follow a company from its ID.

Parameters:

- **following_state_urn** (*str*) – LinkedIn State URN to append to URL to follow the company
- **following** (*bool, optional*) – The following state to set. True by default for following the company

Returns:

Error state. If True, an error occured.

Return type:

boolean

`get_company(public_id)`

Fetch data about a given LinkedIn company.

Parameters:

public_id (*str*) – LinkedIn public ID for a company

Returns:

Company data

Return type:

dict

`get_company_updates(public_id: str | None = None, urn_id: str | None = None, max_results: int | None = None, results: List | None = None) → List`

Fetch company updates (news activity) for a given LinkedIn company.

Parameters:

- **public_id** (*str, optional*) – LinkedIn public ID for a company
- **urn_id** (*str, optional*) – LinkedIn URN ID for a company

Returns:

List of company update objects

Return type:

list

`get_conversation(conversation_urn_id: str)`

Fetch data about a given conversation.

Parameters:

conversation_urn_id (*str*) – LinkedIn URN ID for a conversation

Returns:

Conversation data

Return type:

dict

`get_conversation_details(profile_urn_id)`

Fetch conversation (message thread) details for a given LinkedIn profile.

Parameters:

profile_urn_id (*str*) – LinkedIn URN ID for a profile

Returns:

Conversation data

Return type:

dict

`get_conversations()`

Fetch list of conversations the user is in.

Returns:

List of conversations

Return type:

list

`get_current_profile_views()`

Get profile view statistics, including chart data.

Returns:

Profile view data

Return type:

dict

`get_feed_posts(limit=-1, offset=0, exclude_promoted_posts=True)`

Get a list of URNs from feed sorted by ‘Recent’

Parameters:

- **limit** (*int, optional*) – Maximum length of the returned list, defaults to -1 (no limit)
- **offset** (*int, optional*) – Index to start searching from
- **exclude_promoted_posts** (*bool, optional*) – Exclude from the output promoted posts

Returns:

List of URNs

Return type:

list

`get_invitations(start=0, limit=3)`

Fetch connection invitations for the currently logged in user.

Parameters:

- **start** (*int*) – How much to offset results by
- **limit** (*int*) – Maximum amount of invitations to return

Returns:

List of invitation objects

Return type:

list

`get_job(job_id: str) → Dict`

Fetch data about a given job. :param job_id: LinkedIn job ID :type job_id: str

Returns:

Job data

Return type:

dict

`get_job_skills(job_id: str) → Dict`

Fetch skills associated with a given job. :param job_id: LinkedIn job ID :type job_id: str

Returns:

Job skills

Return type:

dict

`get_post_comments(post_urn: str, comment_count=100) → List`

get_post_comments: Get post comments

Parameters:

- **post_urn** (*str*) – Post URN
- **comment_count** (*int, optional*) – Number of comments to fetch

Returns:

List of post comments

Return type:

list

`get_post_reactions(urn_id, max_results=None, results=None)`

Fetch social reactions for a given LinkedIn post.

Parameters:

- **urn_id** (*str*) – LinkedIn URN ID for a post
- **max_results** (*int, optional*) – Maximum results to return

Returns:

List of social reactions

Return type:

list

# Note: This may need to be updated to GraphQL in the future, see tomquirk/linkedin-api#309

`get_profile(public_id: str | None = None, urn_id: str | None = None) → Dict`

Fetch data for a given LinkedIn profile.

Parameters:

- **public_id** (*str, optional*) – LinkedIn public ID for a profile
- **urn_id** (*str, optional*) – LinkedIn URN ID for a profile

Returns:

Profile data

Return type:

dict

`get_profile_connections(urn_id: str, **kwargs) → List`

Fetch connections for a given LinkedIn profile.

See Linkedin.search_people() for additional searching parameters.

Parameters:

urn_id (*str*) – LinkedIn URN ID for a profile

Returns:

List of search results

Return type:

list

`get_profile_contact_info(public_id: str | None = None, urn_id: str | None = None) → Dict`

Fetch contact information for a given LinkedIn profile. Pass a [public_id] or a [urn_id].

Parameters:

- **public_id** (*str, optional*) – LinkedIn public ID for a profile
- **urn_id** (*str, optional*) – LinkedIn URN ID for a profile

Returns:

Contact data

Return type:

dict

`get_profile_experiences(urn_id: str) → List`

Fetch experiences for a given LinkedIn profile.

NOTE: data structure differs slightly from Linkedin.get_profile() experiences.

Parameters:

urn_id (*str*) – LinkedIn URN ID for a profile

Returns:

List of experiences

Return type:

list

`get_profile_member_badges(public_profile_id: str)`

Fetch badges for a given LinkedIn profile.

Parameters:

public_profile_id (*str*) – public ID of a LinkedIn profile

Returns:

Badges data

Return type:

dict

`get_profile_network_info(public_profile_id: str)`

Fetch network information for a given LinkedIn profile.

Network information includes the following: - number of connections - number of followers - if the account is followable - the network distance between the API session user and the profile - if the API session user is following the profile

Parameters:

public_profile_id (*str*) – public ID of a LinkedIn profile

Returns:

Network data

Return type:

dict

`get_profile_posts(public_id: str | None = None, urn_id: str | None = None, post_count=10) → List` get_profile_posts: Get profile posts

Parameters:

- **public_id** (*str, optional*) – LinkedIn public ID for a profile
- **urn_id** (*str, optional*) – LinkedIn URN ID for a profile
- **post_count** (*int, optional*) – Number of posts to fetch

Returns:

List of posts

Return type:

list

`get_profile_privacy_settings(public_profile_id: str)`

Fetch privacy settings for a given LinkedIn profile.

Parameters:

public_profile_id (*str*) – public ID of a LinkedIn profile

Returns:

Privacy settings data

Return type:

dict

`get_profile_skills(public_id: str | None = None, urn_id: str | None = None) → List`

Fetch the skills listed on a given LinkedIn profile.

Parameters:

- **public_id** (*str, optional*) – LinkedIn public ID for a profile
- **urn_id** (*str, optional*) – LinkedIn URN ID for a profile

Returns:

List of skill objects

Return type:

list

`get_profile_updates(public_id=None, urn_id=None, max_results=None, results=None)`

Fetch profile updates (newsfeed activity) for a given LinkedIn profile.

Parameters:

- **public_id** (*str, optional*) – LinkedIn public ID for a profile
- **urn_id** (*str, optional*) – LinkedIn URN ID for a profile

Returns:

List of profile update objects

Return type:

list

`get_school(public_id)`

Fetch data about a given LinkedIn school.

Parameters:

public_id (*str*) – LinkedIn public ID for a school

Returns:

School data

Return type:

dict

`get_user_profile(use_cache=True) → Dict`

Get the current user profile. If not cached, a network request will be fired.

Returns:

Profile data for currently logged in user

Return type:

dict

`mark_conversation_as_seen(conversation_urn_id: str)`

Send ‘seen’ to a given conversation.

Parameters:

conversation_urn_id (*str*) – LinkedIn URN ID for a conversation

Returns:

Error state. If True, an error occured.

Return type:

boolean

`react_to_post(post_urn_id, reaction_type='LIKE')`

React to a given post. :param post_urn_id: LinkedIn Post URN ID :type post_urn_id: str :param reactionType: LinkedIn reaction type, defaults to “LIKE”, can be “LIKE”, “PRAISE”, “APPRECIATION”, “EMPATHY”, “INTEREST”, “ENTERTAINMENT” :type reactionType: str

Returns:

Error state. If True, an error occured.

Return type:

boolean

`remove_connection(public_profile_id: str)`

Remove a given profile as a connection.

Parameters:

public_profile_id (*str*) – public ID of a LinkedIn profile

Returns:

Error state. True if error occurred

Return type:

boolean

`reply_invitation(invitation_entity_urn: str, invitation_shared_secret: str, action='accept')`

Respond to a connection invitation. By default, accept the invitation.

Parameters:

- **invitation_entity_urn** (*int*) – URN ID of the invitation
- **invitation_shared_secret** (*str*) – Shared secret of invitation
- **action** (*str, optional*) – “accept” or “reject”. Defaults to “accept”

Returns:

Success state. True if successful

Return type:

boolean

`search(params: Dict, limit=-1, offset=0) → List`

Perform a LinkedIn search.

Parameters:

- **params** (*dict*) – Search parameters (see code)
- **limit** (*int, optional*) – Maximum length of the returned list, defaults to -1 (no limit)
- **offset** (*int, optional*) – Index to start searching from

Returns:

List of search results

Return type:

list

`search_companies(keywords: List[str] | None = None, **kwargs) → List`

Perform a LinkedIn search for companies.

Parameters:

keywords (*list, optional*) – A list of search keywords (str)

Returns:

List of companies

Return type:

list

`search_jobs(keywords: str | None = None, companies: List[str] | None = None, experience: List[Literal['1', '2', '3', '4', '5', '6']] | None = None, job_type: List[Literal['F', 'C', 'P', 'T', 'I', 'V', 'O']] | None = None, job_title: List[str] | None = None, industries: List[str] | None = None, location_name: str | None = None, remote: List[Literal['1', '2', '3']] | None = None, listed_at=86400, distance: int | None = None, limit=-1, offset=0, **kwargs) → List[Dict]`

Perform a LinkedIn search for jobs.

Parameters:

- **keywords** (*str, optional*) – Search keywords (str)
- **companies** (*list, optional*) – A list of company URN IDs (str)
- **experience** (*list, optional*) – A list of experience levels, one or many of “1”, “2”, “3”, “4”, “5” and “6” (internship, entry level, associate, mid-senior level, director and executive, respectively)
- **job_type** (*list, optional*) – A list of job types , one or many of “F”, “C”, “P”, “T”, “I”, “V”, “O” (full-time, contract, part-time, temporary, internship, volunteer and “other”, respectively)
- **job_title** (*list, optional*) – A list of title URN IDs (str)
- **industries** (*list, optional*) – A list of industry URN IDs (str)
- **location_name** (*str, optional*) – Name of the location to search within. Example: “Kyiv City, Ukraine”
- **remote** (*list, optional*) – Filter for remote jobs, onsite or hybrid. onsite:”1”, remote:”2”, hybrid:”3”
- **listed_at** (*int/str, optional. Default value is equal to 24 hours.*) – maximum number of seconds passed since job posting. 86400 will filter job postings posted in last 24 hours.
- **distance** (*int/str, optional. If not specified, None or 0, the default value of 25 miles applied.*) – maximum distance from location in miles
- **limit** (*int, optional, default -1*) – maximum number of results obtained from API queries. -1 means maximum which is defined by constants and is equal to 1000 now.
- **offset** (*int, optional*) – indicates how many search results shall be skipped

Returns:

List of jobs

Return type:

list

```
search_people(keywords: str | None = None, connection_of: str | None
= None, network_depths: List[Literal['F', 'S', 'O']] | None = None,
current_company: List[str] | None = None, past_companies: List[str] |
None = None, nonprofit_interests: List[str] | None = None,
profile_languages: List[str] | None = None, regions: List[str] | None
= None, industries: List[str] | None = None, schools: List[str] | None
= None, contact_interests: List[str] | None = None,
service_categories: List[str] | None = None,
include_private_profiles=False, keyword_first_name: str | None = None,
keyword_last_name: str | None = None, keyword_title: str | None =
None, keyword_company: str | None = None, keyword_school: str | None =
None, network_depth: Literal['F'] | Literal['S'] | Literal['O'] | None
= None, title: str | None = None, **kwargs) → List[Dict]
```

Perform a LinkedIn search for people.

Parameters:

- **keywords** (*str, optional*) – Keywords to search on
- **current_company** (*list, optional*) – A list of company URN IDs (str)
- **past_companies** (*list, optional*) – A list of company URN IDs (str)
- **regions** (*list, optional*) – A list of geo URN IDs (str)
- **industries** (*list, optional*) – A list of industry URN IDs (str)
- **schools** (*list, optional*) – A list of school URN IDs (str)
- **profile_languages** (*list, optional*) – A list of 2-letter language codes (str)
- **contact_interests** (*list, optional*) – A list containing one or both of "proBono" and "boardMember"
- **service_categories** (*list, optional*) – A list of service category URN IDs (str)
- **network_depth** (*str, optional*) – Deprecated, use *network_depths*. One of "F", "S" and "O" (first, second and third+ respectively)
- **network_depths** (*list, optional*) – A list containing one or many of "F", "S" and "O" (first, second and third+ respectively)
- **include_private_profiles** (*boolean, optional*) – Include private profiles in search results. If False, only public profiles are included. Defaults to False
- **keyword_first_name** (*str, optional*) – First name
- **keyword_last_name** (*str, optional*) – Last name
- **keyword_title** (*str, optional*) – Job title
- **keyword_company** (*str, optional*) – Company name
- **keyword_school** (*str, optional*) – School name
- **connection_of** (*str, optional*) – Connection of LinkedIn user, given by profile URN ID
- **limit** (*int, optional*) – Maximum length of the returned list, defaults to -1 (no limit)

Returns:

List of profiles (minimal data only)

Return type:

list

`send_message(message_body: str, conversation_urn_id: str | None = None, recipients: List[str] | None = None)`

Send a message to a given conversation.

Parameters:

- **message_body** (*str*) – Message text to send
- **conversation_urn_id** (*str, optional*) – LinkedIn URN ID for a conversation
- **recipients** (*list, optional*) – List of profile urn id's

Returns:

Error state. If True, an error occured.

Return type:

boolean

`unfollow_entity(urn_id: str)`

Unfollow a given entity.

Parameters:

urn_id (*str*) – URN ID of entity to unfollow

Returns:

Error state. Returns True if error occurred

Return type:

boolean