LinkedIn API Overview

LinkedIn's home for API documentation for all LinkedIn business lines. Our API documentation is organized by business lines covering Consumer, Compliance, Learning, Marketing, Sales, and Talent Solutions. Follow the links below to learn more about business lines and their possible integration types.

Where to Start

GET STARTED

LinkedIn Business Solutions

Get API Access

Authentication

API Concepts

Breaking Change Policy

Best Practices

Error Handling

Consumer Solutions

OVERVIEW

Consumer Overview

Sign in with LinkedIn using OpenID Connect

Share on LinkedIn

Plugins

Learning Solutions

OVERVIEW

Learning Home

Learning Overview

Request Access

API Terminology

API Foundations

Marketing Solutions

OVERVIEW

Marketing Overview

Getting Started

Integrations Overview

Apply for Access

Recent Changes

Talent Solutions

OVERVIEW

Talent Overview

Apply Connect

Apply with LinkedIn

Easy Apply

Job Posting

Recruiter System Connect

CRM Connect

Sales Solutions

OVERVIEW

Sales Overview

Analytics Services

Display Services

Sync Services

Compliance

OVERVIEW

Compliance Overview

Release Notes

Getting Access to LinkedIn APIs

06/26/2025

The LinkedIn API uses OAuth 2.0 for user authorization and API authentication. Applications must be authorized and authenticated before they can fetch data from LinkedIn or get access to member data. This page summarizes the available permissions and partner programs available for accessing LinkedIn APIs. Most permissions and partner programs require explicit approval from LinkedIn. Open Permissions are the only permissions that are available to all developers without special approval.

All permissions listed below are used in either Member Authentication Flow (3-legged) or Application Authentication Flow (2-legged). More about these permission types can be found in Authenticating with OAuth 2.0 Overview.

Open Permissions (Consumer)

The following permissions are available to all developers, and may be added via self-service through the LinkedIn Developer Portal, under the Products tab on your application page. LinkedIn products can be enabled after creating a new application.

Expand table

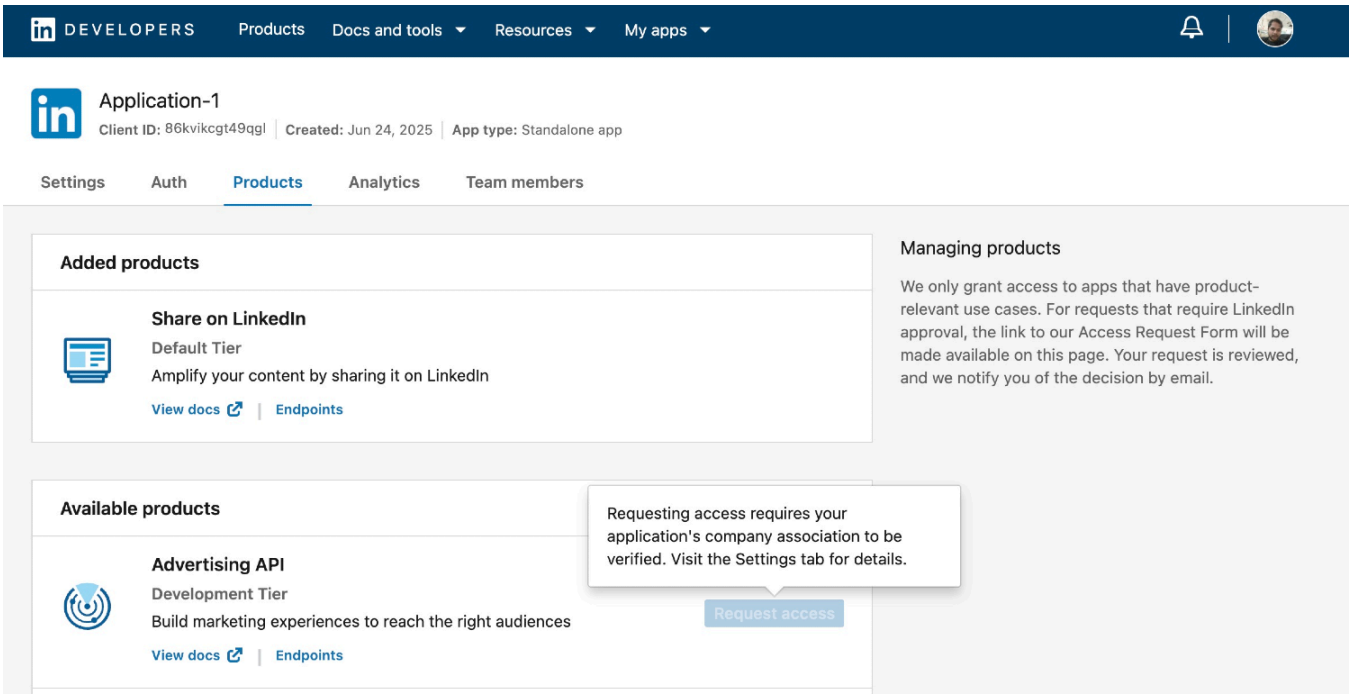
Product/Program	Permission	Description
Sign in with LinkedIn using OpenID Connect	profile	**Member Auth**: Retrieve authenticated member's name, headline, and photo.
Sign in with LinkedIn using OpenID Connect	email	**Member Auth**: Retrieve authenticated member's primary email address.
Share on LinkedIn	w_member_social	**Member Auth**: Post, comment and like posts on behalf of an authenticated member.

Learning

Developers seeking to build a learning related integration should refer to the Request API Access page within the LinkedIn Learning API space.

Marketing

Developers seeking to build a marketing related integration using Advertising API permissions must be approved. You can apply for access through the Developer Portal. To do this, select your app from My Apps, navigate to the Products tab, and add the Advertising API product. More information of LinkedIn Marketing Partners here.



Audiences permissions may be applied for after becoming an approved Advertising API partner. Contact support or your partner representative for application information.

For the most up-to-date list of available permissions and their descriptions, see the LinkedIn Marketing API Permissions Table.

Sales

Developers seeking to build sales related integration using one of the permissions below must be approved as a Sales Navigator Application Platform (SNAP) partner. Apply here to be a SNAP partner.

Expand table

Product/Program	Permission	Description
Sales Navigator Application Platform(SNAP)	r_sales_nav_analytics	**Member Auth**: Enables access to Sales Navigator Analytics retrieval.
Sales Navigator Application Platform(SNAP)	r_sales_nav_display	**Member Auth**: Display Services permission for Sales Navigator.
Sales Navigator Application Platform(SNAP)	r_sales_nav_validation	**Application Auth**: Access Sales Navigator endpoints for CRM data validation.

Product/Program	Permission	Description
Sales Navigator Application Platform(SNAP)	r_sales_nav_profiles	**Application Auth**: Access Sales Navigator endpoints that present matched, publicly available member profile information.

Talent

Developers seeking to build talent related integrations through one of the programs listed below can apply here. We recommend familiarizing yourself with the types of partner integrations available before applying by visiting here and here.

- Recruiter System Connect (RSC)
- Apply Connect
- Apply with LinkedIn
- Premium Job Posting
- Easy Apply

Compliance (Closed)

The following permissions used for Compliance integrations are listed for reference purposes only. Access is closed and may not be requested.

Expand table

Product/Program	Permission	Description
Compliance	r_compliance	**Member Auth**: Retrieve activities for compliance monitoring and archiving
Compliance	w_compliance	**Member Auth**: Manage and delete data for compliance.

Overview

Article • 05/08/2023

The LinkedIn API uses OAuth 2.0 for member (user) authorization and API authentication. Applications must be authorized and authenticated before they can fetch data from LinkedIn or get access to LinkedIn member data.

There are two types of Authorization Flows available:

- **Member Authorization (3-legged OAuth)**
- **Application Authorization (2-legged OAuth)**

Depending on the type of permissions your integration will require, follow one of the authorization flows to get started.

> **Note**
>
> - There are several third-party libraries in the open source community which abstract the OAuth 2.0 authentication process in every major programming language.
> - LinkedIn does not support TLS 1.0.

Member Authorization (3-legged OAuth Flow)

The Member Authorization grants permissions to your application by a LinkedIn member to access the member's resources on LinkedIn. Your application has no access to these resources without member approval. The Member Auth uses the 3-legged OAuth code flow. For step-by-step instructions on how to implement 3-legged OAuth, see **Authorization Code Flow (3-legged OAuth)** page.

> **Tip**
>
> **When to use 3-legged OAuth**
> Use this flow if you are requesting access to a member's account to use their data and make requests on their behalf. This is the most commonly used permission type across LinkedIn APIs. Open permissions available to all applications are of this type such as `r_liteprofile`, `r_emailaddress`, and `w_member_social`.

Member Auth Permissions

Member Authorization Permissions are granted by a LinkedIn member to access members resources on LinkedIn. Permissions are authorization consents to access LinkedIn resources. The LinkedIn platform uses permissions to protect and prevent abuse of member data. Your application must have the appropriate permissions before it can access data. To see the list of permissions, descriptions and access details, refer to Getting Access to LinkedIn APIs page.

Application Authorization (2-legged OAuth Client Credential Flow)

Application Authorization or using 2-Legged OAuth grants permissions to your application to access protected LinkedIn resources. If you are accessing APIs that are not member specific, use this flow. Not all APIs support Application Authorization. For example, Marketing APIs you must use Member Authorization explained above. For step-by-step instructions on how to implement 2-legged OAuth, see **Client Credential Flow (2-legged OAuth)** page.

> ⓘ **Note**
>
> Always request the minimal permission scopes necessary for your use case.

Application Auth Permissions

Application Authorization Permissions are granted to applications to access LinkedIn protected resources. To see the list of permissions, descriptions and access details, refer to Getting Access to LinkedIn APIs page.

Sample Application

You can explore the OAuth Sample Applications that enables you to try out RESTful OAuth calls to the LinkedIn Authentication server. The sample app is available in Java.

Additionally, you can also explore the Marketing Sample Application.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Authorization Code Flow (3-legged OAuth)

10/15/2025

The Authorization Code Flow is used for applications to request permission from a LinkedIn member to access their account data. The level of access or profile detail is explicitly requested using the `scope` parameter during the authorization process outlined below. This workflow will send a consent prompt to a selected member, and once approved your application may begin making API calls on behalf of that member.

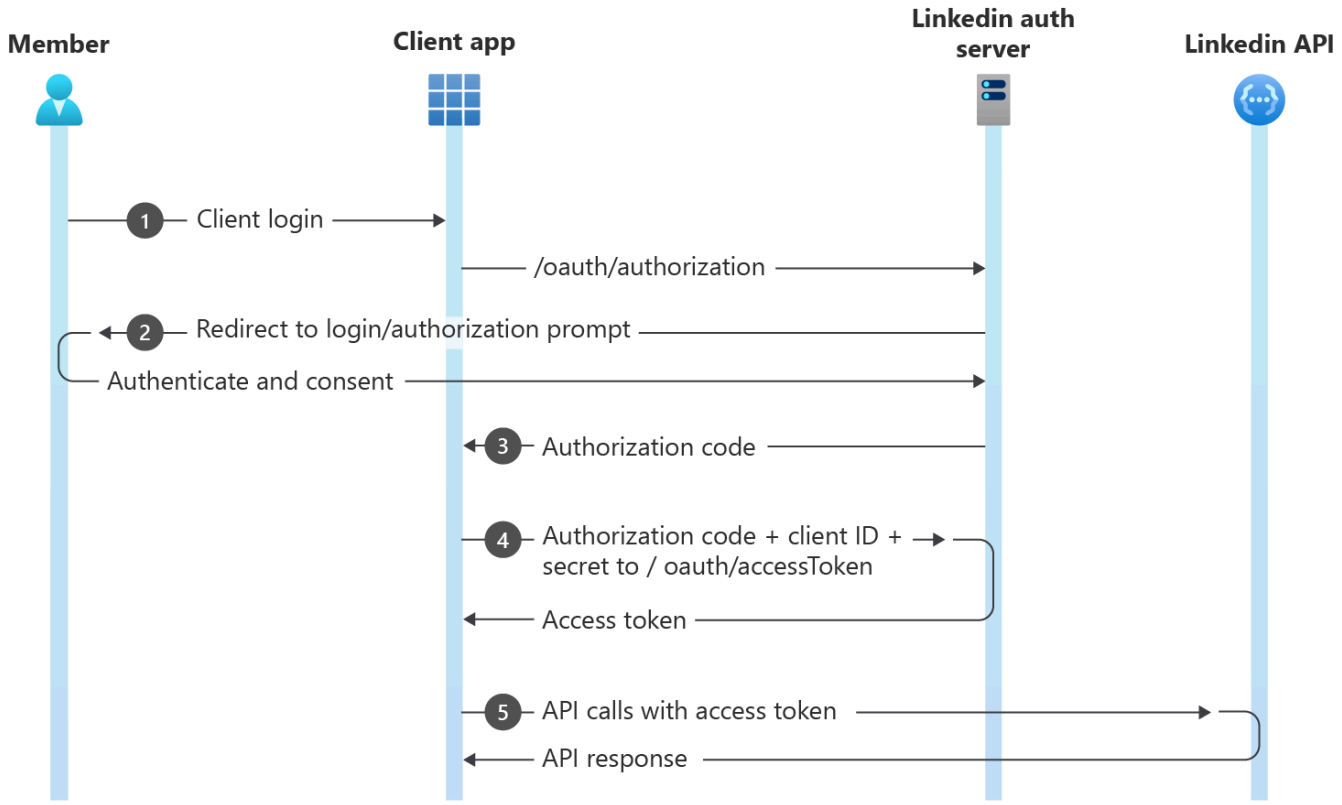
This approval process ensures that LinkedIn members are aware of what level of detail an application may access or action it may perform on their behalf.

If multiple scopes are requested, the user must consent to all of them and may not select the individual scopes. For the benefit of your LinkedIn users, please ensure that your application requests the least number of scope permissions.

> **Note**
>
> **Generate a Token Manually Using the LinkedIn Developer Portal**
> The LinkedIn Developer Portal has a token generator for manually creating tokens. Visit the LinkedIn Developer Portal Token Generator or follow the steps outlined in LinkedIn Developer Portal Tools.

Authorization Code Flow

1. Configure your application in the Developer Portal to obtain *Client ID* and *Client Secret*.
2. Your application directs the browser to LinkedIn's OAuth 2.0 authorization page where the member authenticates.
3. After authentication, LinkedIn's authorization server passes an authorization code to your application.
4. Your application sends this code to LinkedIn and LinkedIn returns an access token.
5. Your application uses this token to make API calls on behalf of the member.



How to Implement 3-legged OAuth

Follow the steps given below to implement the 3-legged OAuth for LinkedIn APIs:

Prerequisites

- A LinkedIn Developer application to create a new application or select your existing application
- Prior authorization access granted for at least one 3-legged OAuth permission.

The permission request workflow is outlined in the Getting Access section.

Step 1: Configure Your Application

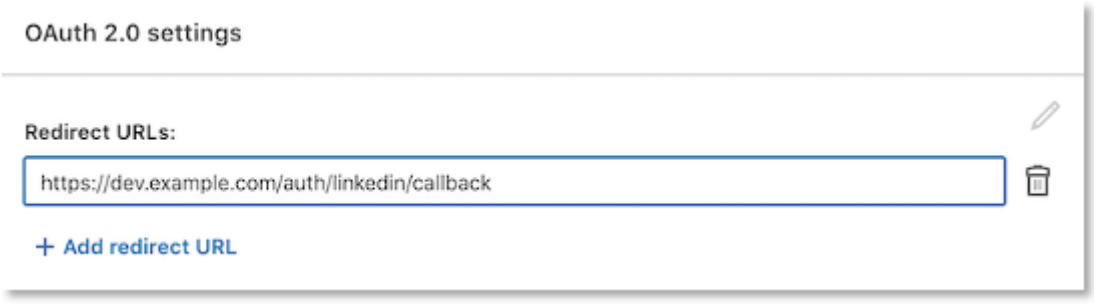
1. Select your application in the LinkedIn Developer Portal.
2. Click the **Auth** tab to view your application credentials.
3. Add the redirect (callback) URL via `HTTPS` to your server.

> **Note**
>
> LinkedIn servers will only communicate with URLs that you have identified as trusted.
>
> - URLs must be absolute:

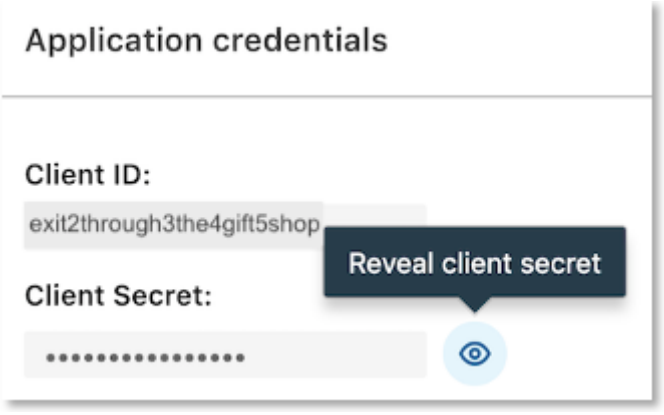
- `https://dev.example.com/auth/linkedin/callback`
- *not* `/auth/linkedin/callback`

- Parameters are ignored:
  - `https://dev.example.com/auth/linkedin/callback?id=1`
  - *will be* `https://dev.example.com/auth/linkedin/callback`
- URLs cannot include a #
  - `https://dev.example.com/auth/linkedin/callback#linkedin` is invalid.

If you are using Postman to test this flow, use `https://oauth.pstmn.io/v1/callback` as your redirect URL and enable **Authorize using browser**.



Each application is assigned a unique **Client ID** (Consumer key/API key) and **Client Secret**. Please make a note of these values as they will be integrated into your application. Your **Client Secret** protects your application's security so be sure to keep it secure!



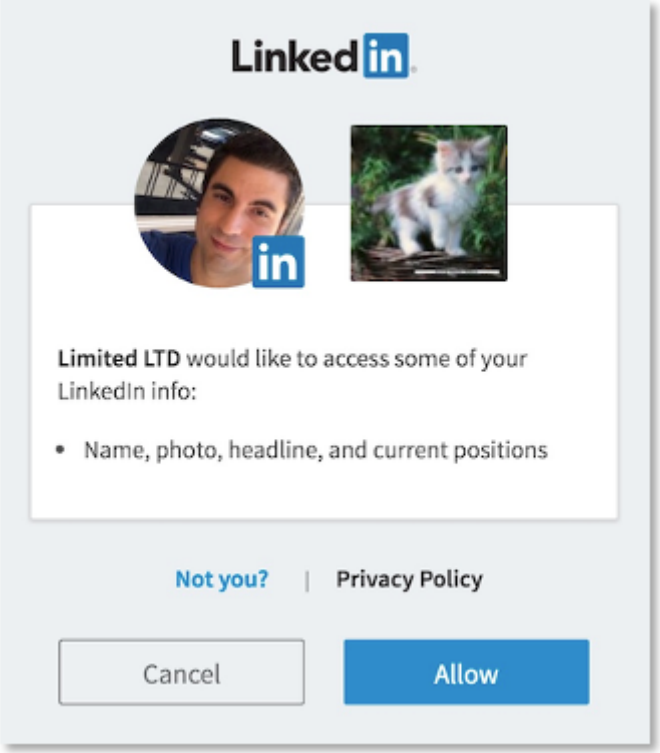
> ⚠ **Warning**
>
> Do not share your *Client Secret* value with anyone, and **do not** pass it in the URL when making API calls, or URI query-string parameters, or post in support forums, chat, etc.

Step 2: Request an Authorization Code

To request an authorization code, you must direct the member's browser to LinkedIn's OAuth 2.0 authorization page, where the member either accepts or denies your application's permission request. Ensure that you follow the standard OAuth specification when invoking OAuth in your app.

Once the request is made, one of the following occurs:

1. If it is a first-time request, the permission request timed out, or was manually revoked by the member: the browser is redirected to LinkedIn's authorization consent window.
2. If there is an existing permission grant from the member: the authorization screen is bypassed and the member is immediately redirected to the URL provided in the `redirect_uri` query parameter.



When the member completes the authorization process, the browser is redirected to the URL provided in the `redirect_uri` query parameter.

> **Note**
>
> If the `scope` permissions are changed in your app, your users must re-authenticate to ensure that they have explicitly granted your application all of the permissions that it is requesting on their behalf.

https

```
GET https://www.linkedin.com/oauth/v2/authorization
```

Expand table

Parameter	Type	Description	Required
response_type	string	The value of this field should always be: `code`	Yes
client_id	string	The *API Key* value generated when you registered your application.	Yes
redirect_uri	url	The URI your users are sent back to after authorization. This value must match one of the *Redirect URLs* defined in your application configuration. For example, `https://dev.example.com/auth/linkedin/callback`.	Yes
state	string	A unique string value of your choice that is hard to guess. Used to prevent CSRF. For example, `state=DCEeFWf45A53sdfKef424`.	No
scope	string	URL-encoded, space-delimited list of member permissions your application is requesting on behalf of the user. These must be explicitly requested. For example, `scope=liteprofile%20emailaddress%20w_member_social`. See Permissions and Best Practices for Application Development for additional information.	Yes

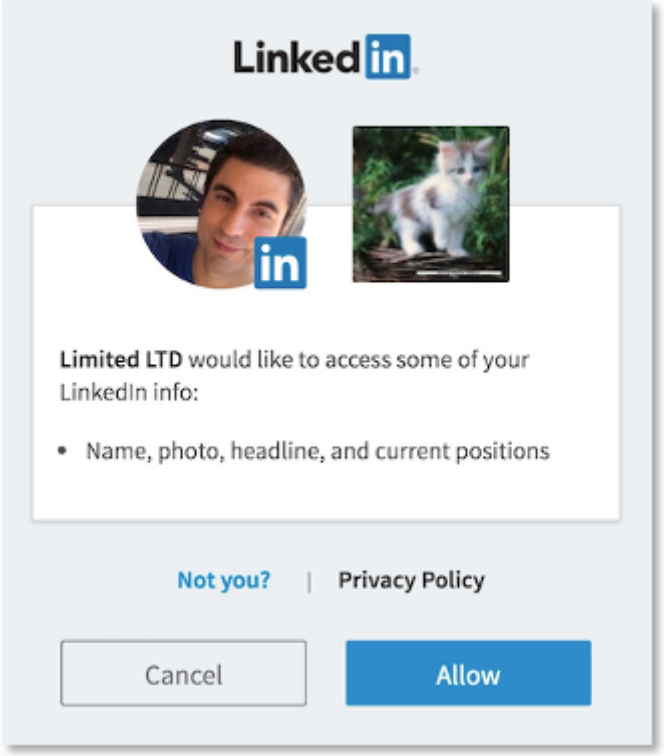
The scopes available to your app depend on which Products or Partner Programs your app has access to. This information is available in the LinkedIn Developer Portal. Your app's Auth tab will show current scopes available. You can apply for new Products under the Products tab. If approved, your app will have access to new scopes.

Sample Request

https

```
GET https://www.linkedin.com/oauth/v2/authorization?response_type=code&client_id=
{your_client_id}&redirect_uri=
{your_callback_url}&state=foobar&scope=liteprofile%20emailaddress%20w_member_socia
l
```

Once redirected, the member is presented with LinkedIn's authentication screen. This identifies your application and outlines the particular member permissions/scopes that your application is requesting. You can change the logo and application name in the LinkedIn Developer Portal under My apps > Settings.



Member Approves Request

By providing valid LinkedIn credentials and clicking **Allow**, the member approves your application's request to access their member data and interact with LinkedIn on their behalf. This approval instructs LinkedIn to redirect the member to the redirect URL that you defined in your `redirect_uri` parameter.

https

```
https://dev.example.com/auth/linkedin/callback?
state=foobar&code=AQTQmah11lalyH65DAIivsjsAQV5P-
1VTVVebnLl_SCiyMXoIjDmJ4s6rO1VBGP5Hx2542KaR_eNawkrWiCiAGxIaV-TCK-
mkxDISDak08tdaBzgUYfnTJL1fHRoDWCcC2L6LXBCR_z2XHzeWSuqTkR1_jO8CeV9E_WshsJBgE-
PWElyvsmfuEXLQbCLfj8CHasuLafFpGb0glO4d7M
```

Attached to the `redirect_uri` are two important URL arguments that you need to read from the request:

- `code` — The OAuth 2.0 authorization code.
- `state` — A value used to test for possible CSRF attacks.

The `code` is a value that you exchange with LinkedIn for an OAuth 2.0 access token in the next step of the authentication process. For security reasons, the authorization code has a 30-minute lifespan and must be used immediately. If it expires, you must repeat all of the previous steps to request another authorization code.

> ⚠ **Warning**
>
> Before you use the authorization code, your application should ensure that the value returned in the `state` parameter matches the `state` value from your original authorization code request. This ensures that you are dealing with the real member and not a malicious script. If the state values do not match, you are likely the victim of a CSRF attack and your application should return a `401 Unauthorized` error code in response.

Failed Requests

If the member chooses to cancel, or the request fails for any reason, the client is redirected to your `redirect_uri` with the following additional query parameters appended:

- `error` - A code indicating one of these errors:
  - `user_cancelled_login` - The member declined to log in to their LinkedIn account.
  - `user_cancelled_authorize` - The member refused to authorize the permissions request from your application.
- `error_description` - A URL-encoded textual description that summarizes the error.
- `state` - A value passed by your application to prevent CSRF attacks.

For more error details, see here

Step 3: Exchange Authorization Code for an Access Token

The next step is to get an access token for your application using the authorization code from the previous step.

```
https

POST https://www.linkedin.com/oauth/v2/accessToken
```

To do this, make the following HTTP POST request with a `Content-Type` header of `x-www-form-urlencoded` using the following parameters:

Expand table

Parameter	Type	Description	Required
grant_type	string	The value of this field should always be: `authorization_code`	Yes
code	string	The authorization code you received in Step 2.	Yes
client_id	string	The Client ID value generated in Step 1.	Yes
client_secret	string	The Secret Key value generated in Step 1. See the Best Practices Guide for ways to keep your `client_secret` value secure.	Yes
redirect_uri	url	The same `redirect_uri` value that you passed in the previous step.	Yes

Sample Request

https

https

```
POST  https://www.linkedin.com/oauth/v2/accessToken

Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
code={authorization_code_from_step2_response}
client_id={your_client_id}
client_secret={your_client_secret}
redirect_uri={your_callback_url}
```

Response

A successful access token request returns a JSON object containing the following fields:

Expand table

Parameter	Type	Description
access_token	string	The access token for the application. This value must be kept secure as specified in the API Terms of Use. The length of access tokens is ~500 characters. We recommend that you plan for your application to handle tokens with length of at least 1000 characters to accommodate any future expansion plans. This applies to both access tokens and refresh tokens.
expires_in	int	The number of seconds remaining until the token expires. Currently, all access tokens are issued with a 60-day lifespan.

Parameter	Type	Description
refresh_token	string	Your refresh token for the application. This token must be kept secure.
refresh_token_expires_in	int	The number of seconds remaining until the refresh token expires. Refresh tokens usually have a longer lifespan than access tokens.
scope	string	URL-encoded, space-delimited list of member permissions your application has requested on behalf of the user.

JSON

```
{
"access_token":"AQUvlL_DYEzvT2wz1QJiEPeLioeA",
"expires_in":5184000,
"scope":"r_basicprofile"
}
```

For more error details, refer to the API Error Details table.

> ⓘ Note
>
> **Access Token Scopes and Lifetime**
> Access tokens stay valid until the number of seconds indicated in the `expires_in` field in the API response. You can go through the OAuth flow on multiple clients (browsers or devices) and simultaneously hold multiple valid access tokens if the same scope is requested. If you request a different scope than the previously granted scope, all the previous access tokens are invalidated.

Step 4: Make Authenticated Requests

Once you've obtained an access token, you can start making authenticated API requests on behalf of the member by including an `Authorization` header in the HTTP call to LinkedIn's API.

Sample Request

Bash

```
curl -X GET 'https://api.linkedin.com/v2/me' \
-H 'Authorization: Bearer {INSERT_TOKEN}'
```

Step 5: Refresh Access Token

> **Tip**
>
> To protect members' data, LinkedIn does not generate long-lived access tokens.
>
> > Make sure your application refreshes access tokens before they expire, to avoid unnecessarily sending your application's users through the authorization process again.

Refreshing an access token is a seamless user experience. To refresh an access token, go through the authorization process again to fetch a new token. This time however, in the refresh workflow, the authorization screen is bypassed, and the member is redirected to your redirect URL, provided the following conditions are met:

- The member is still logged into www.linkedin.com
- The member's current access token has not expired

If the member is no longer logged in to www.linkedin.com or their access token has expired, they are sent through the normal authorization process.

Programmatic refresh tokens are available for a limited set of partners. If this feature has been enabled for your application, see Programmatic Refresh Tokens for instructions.

API Error Details

Following are the API errors and thier resolution for 3-legged OAuth. If you wish to view the standard HTTP status codes and thier meaning, see Error Handling page.

/oauth/v2/authorization

Expand table

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
401	Redirect_uri doesn't match	Redirect URI passed in the request does not match the redirect URI added to the developer application.	Ensure that the redirect URI passed in the request match the redirect URI added in the developer application under the Authorization tab.

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
401	Client_id doesn't match	Client ID passed in the request does not match the client ID of the developer application.	Ensure that the client ID passed is in match with the developer application.
401	Invalid scope	Permissions passed in the request is invalid	Ensure that the permissions sent in scope parameter is assigned to the developer application in the LinkedIn developer portal.

/oauth/v2/accessToken

Expand table

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
401	invalid_request "Unable to retrieve access token: authorization code not found"	Authorization code sent is invalid or not found.	Check whether the sent authorization code is valid.
400	invalid_request "A required parameter "redirect_uri" is missing"	Redirect_uri in the request is missing. It is a mandatory parameter.	Pass the redirect_uri in the request to route user back to correct landing page.
400	invalid_request "A required parameter "code" is missing"	Authorization code in the request is missing. It is a mandatory parameter.	Pass the Authorization code received as part of authorization API call.
400	invalid_request "A required parameter "grant_type" is missing"	Grant type in the request is missing. It is a mandatory parameter.	Add grant_type as "authorization_code" in the request.
400	invalid_request "A required parameter "client_id" is missing"	Client ID in the request is missing. It is a mandatory parameter.	Pass the client id of the app in request.

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
400	invalid_request "A required parameter "client_secret" is missing"	Client Secret in the request is missing. It is a mandatory parameter.	Pass the client secret of the app in request.
400	invalid_redirect_uri "Unable to retrieve access token: appid/redirect uri/code verifier does not match authorization code. Or authorization code expired. Or external member binding exists"	Invalid redirect uri is passed in the request.	Pass the right redirect uri tagged to the developer application.
400	invalid_redirect_uri "Unable to retrieve access token: appid/redirect uri/code verifier does not match authorization code. Or authorization code expired. Or external member binding exists"	Invalid redirect uri is passed in the request.	Pass the right redirect uri tagged to the developer application.
400	invalid_redirect_uri "Unable to retrieve access token: appid/redirect uri/code verifier does not match authorization code. Or authorization code expired. Or external member binding exists"	Invalid Authorization code is sent as part of the request.	Authorization code expired; re-authenticate member to generate new authorization code and pass the fresh authorization code to exchange for access token.

Client Credential Flow (2-legged OAuth)

Article • 02/05/2025

If your application needs to access APIs that are not member specific, use the Client Credential Flow. Your application **cannot** access these APIs by default.

Learn more:

- LinkedIn Developer Enterprise products and permission requests.
- LinkedIn Developers Platform knowledge base.

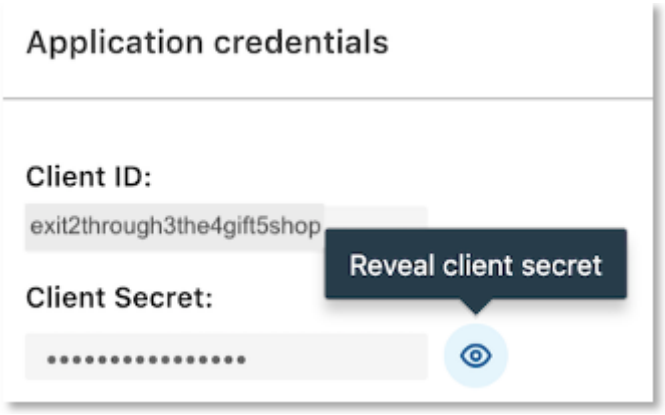
> **Important**
>
> 2-legged OAuth authentication is not available for Marketing APIs

> **Note**
>
> **Generate a Token Manually Using the Developer Portal**
> The LinkedIn Developer Portal has a token generator for manually creating tokens. Visit the LinkedIn Developer Portal Token Generator or follow the steps outlined in Developer Portal Tools.

Step 1: Get Client ID and Client Secret

- Getting started? Create a new application on the Developer Portal.
- Existing application? Go to My apps to modify your app settings.

Each application is assigned a unique *Client ID* (Consumer key/API key) and *Client Secret*. Please make a note of these values as they will be integrated into your application config files. Your *Client Secret* protects your application's security so be sure to keep it secure!



> ⚠ **Warning**
>
> Do not share your *Client Secret* value with anyone, and **do not** pass it in the URL when making API calls, or URI query-string parameters, or post in support forums, chat, etc.

Step 2: Generate an Access Token

To generate an access token, issue a HTTP POST against `accessToken` with a `Content-Type` header of `x-www-form-urlencoded` and the following parameters in the request body:

```
https
```

```
https://www.linkedin.com/oauth/v2/accessToken
```

Expand table

Parameter	Description	Required
grant_type	The value of this field should always be `client_credentials`	Yes
client_id	The *Client ID* value generated when you registered your application	Yes
client_secret	The *Client Secret* value generated when you registered your application. All values requiring URL encoding must be encoded. Client secrets can include characters like `/`, `=`, `+` which require URL encoding.	Yes

- View the Best Practices for Secure Applications page for more security info.

Sample Request (Secure Approach)

https

```
https

POST https://www.linkedin.com/oauth/v2/accessToken HTTP/1.1

Content-Type: application/x-www-form-urlencoded
grant_type=client_credentials
client_id={your_client_id}
client_secret={your_client_secret}
```

A successful access token request returns a JSON object containing the following fields:

- `access_token` — The access token for the application. This token must be kept secure.
- `expires_in` — Seconds until token expiration.
  - The access token has a 30-minute lifespan and must be used immediately. You may request a new token once your current token expires.

Sample Response

```
JSON

{
    "access_token": "AQV8...",
    "expires_in": "1800"
}
```

For error details, refer the API Error Details section.

Step 3: Make API Requests

Once you've received an access token, you can make API requests by including an *Authorization* header with your token in the HTTP call to LinkedIn's API.

Sample Request

```
https

GET https://api.linkedin.com/v2/jobs HTTP/1.1
```

```
Connection: Keep-Alive
Authorization: Bearer {access_token}
```

API Error Details

Expand table

HTTP STATUS CODE	ERROR MESSAGE	DESCRIPTION	RESOLUTION
401	invalid_client_id "Client authentication failed"	Client Authentication failed due to bad client credentials passed as part of the request.	Check whether the right **Client ID**, **Client Secret** are passed as part of the request.
401	access_denied "This application is not allowed to create application tokens"	The developer application doesn't have enough permission to generate 2L application token.	Reach out to the LinkedIn Relationship Manager or Business Development team to get the necessary access.
400	invalid_request "A required parameter "grant_type" is missing"	Grant type in the request is missing. It is a mandatory parameter.	Add grant_type as `client_credentials` in the request.
400	invalid_request "A required parameter "client_id" is missing"	Client ID in the request is missing. It is a mandatory parameter.	Pass the Client ID of the developer application in request.
400	invalid_request "A required parameter "client_secret" is missing"	Client Secret in the request is missing. It is a mandatory parameter.	Pass the Client Secret of the developer application in the request.
400	invalid_client_id "The passed in client_id is invalid "abcdefghijk""	Invalid client ID is passed in the request.	Pass the right client ID from the developer application.

Feedback

Was this page helpful? Yes No

Provide product feedback

Generate an Access Token - Getting Started with Postman

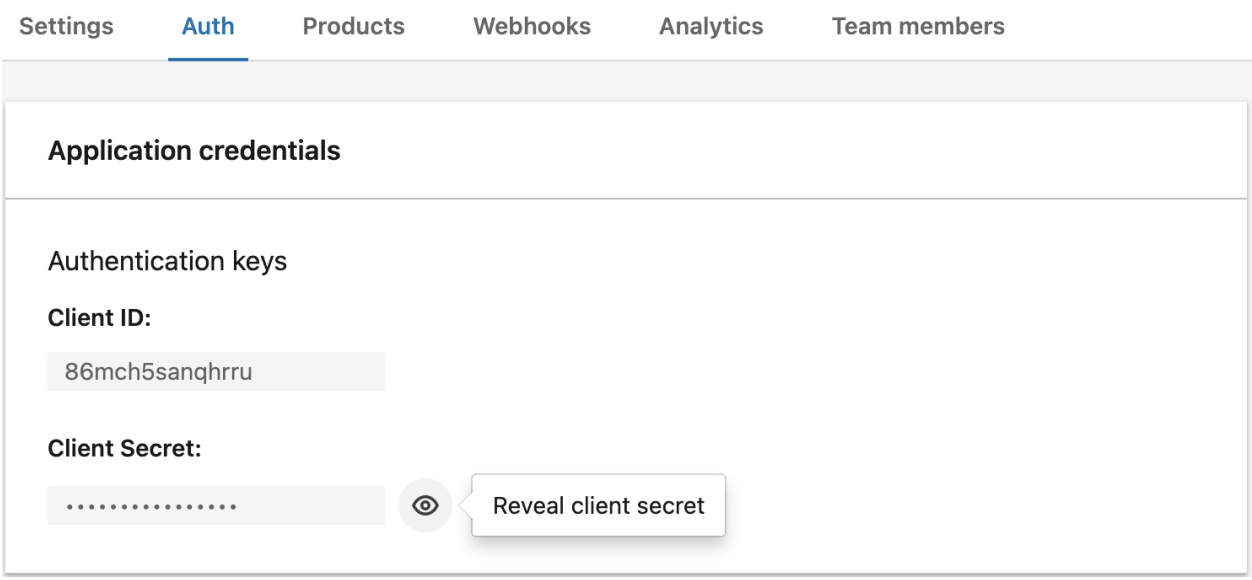
Article • 05/08/2023

Summary

The full process your application will need to implement for 3-legged tokens is described in Authorization Code Flow and 2-legged tokens is described in Client Credentials Flow. The steps outlined below describe the process for using LinkedIn's Public Postman workspaces to generate OAuth tokens for testing. For any specific examples, we will use the Marketing Solutions workspace, but all steps should easily apply to all workspaces. These steps assume you have already created a free Postman account .

Step 1 - Application

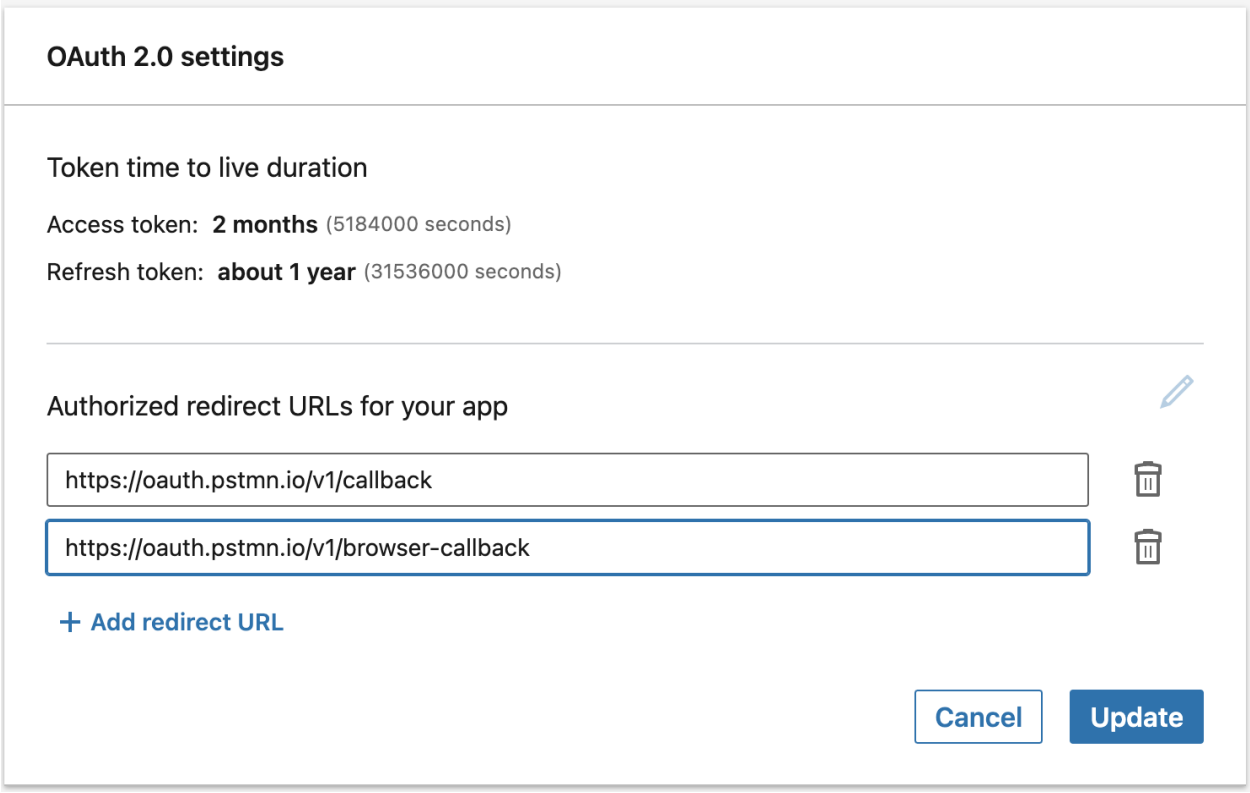
Go to the LinkedIn Developer Portal , select the app you'll be using, click the "Auth" tab, and locate your Client ID and Client Secret. Please note these values for use later during this process.



Step 2 - Auth Settings

From the same "Auth" tab, scroll to the bottom of the page. Under "OAuth 2.0 Settings", add the Postman callback URLs `https://oauth.pstmn.io/v1/callback` and `https://oauth.pstmn.io/v1/browser-callback` to your Redirect URL list.

> ⊗ **Caution**
>
> Postman uses the term "Callback URL"
> LinkedIn uses the term "Redirect URL"



Step 3 - Fork Collections and Environments

Navigate to LinkedIn's public Postman workspaces:

Choose a workspace and fork the collections and relevant environments of interest. Each collection will have an environment it should be used with. For example, if you were to navigate to the LinkedIn Marketing Solutions workspace, the Campaign Management collection should be used with the `campaign-management-env` environment.

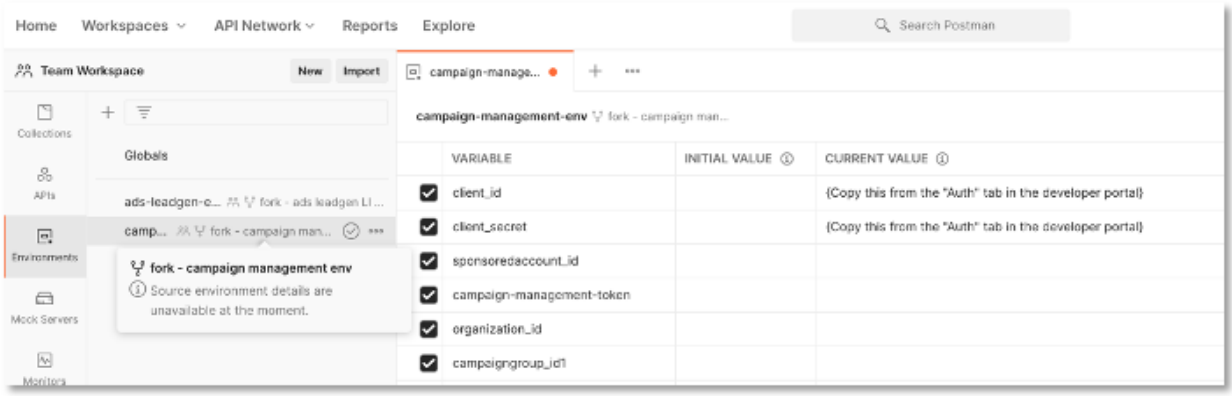
Fork a Postman Collection

Fork a collection:
https://www.microsoft.com/en-us/videoplayer/embed/RWNqGu?postJsllMsg=true

Fork an environment:
https://www.microsoft.com/en-us/videoplayer/embed/RWNqGv?postJsllMsg=true

Step 4 - Fill in Environment Variables

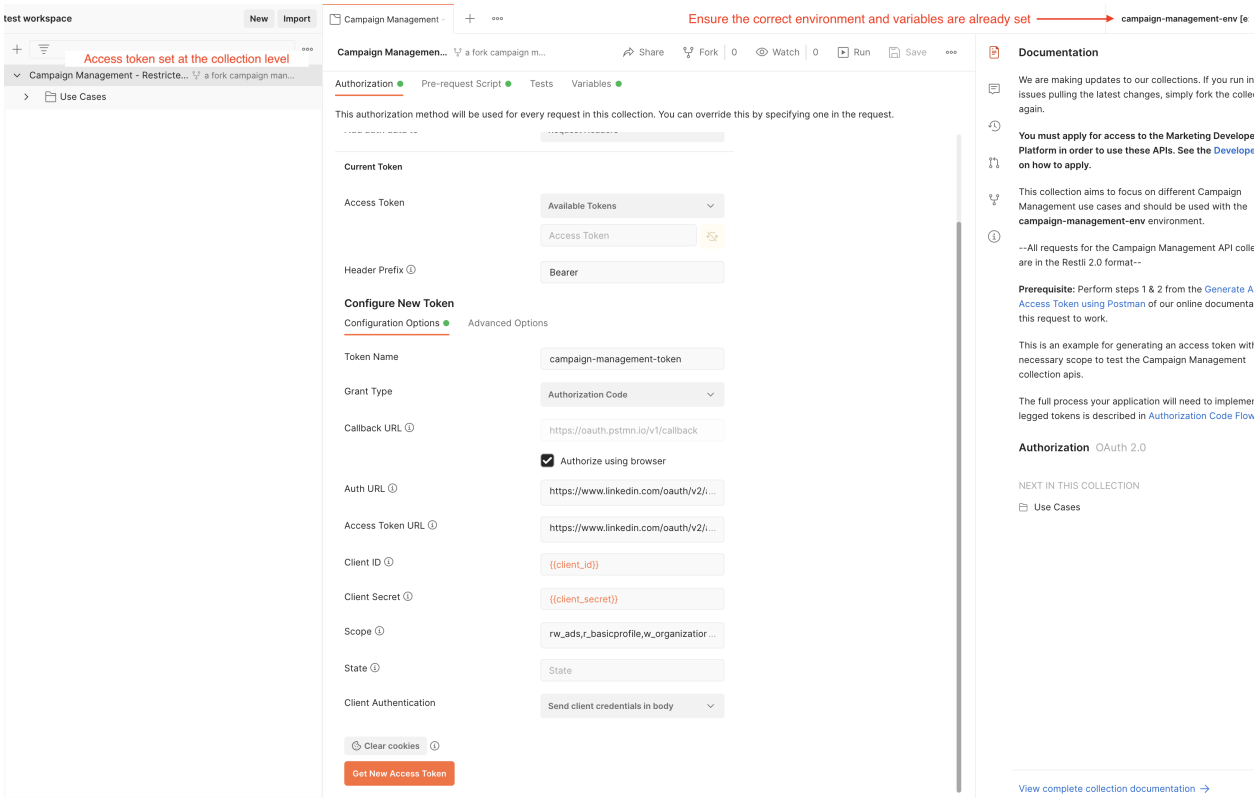
Fill in the Client ID and Client Secret environment variables before moving onto the next step. Don't forget to save your changes!



Step 5 - Headers

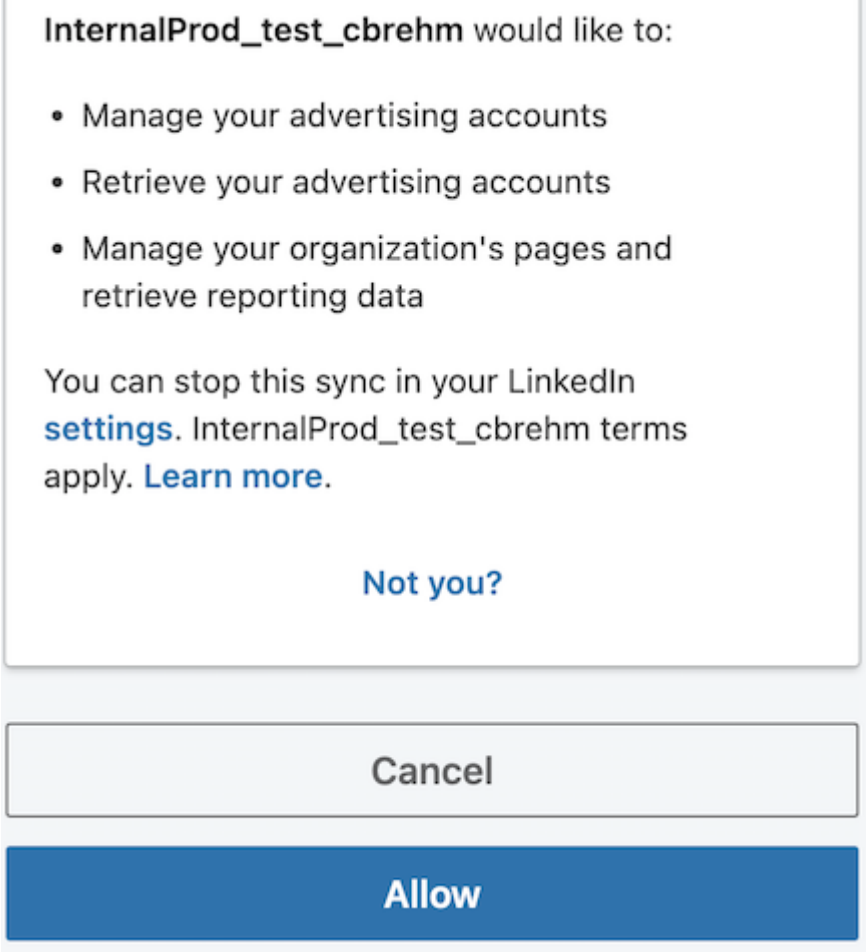
Each collection in each workspace will have its OAuth 2.0 Authorization settings pre-populated with the correct URLs, environment variables, and scopes to be able to successfully run the requests within the corresponding Use Cases folder. Click on a collection title to open it's Authorization tab. Ensure that the correct environment is selected and click "Get new access token":

- Grant Type: Authorization Code (3-legged token) or Client Credentials (2-legged token)
- Callback (Redirect) URL: `https://oauth.pstmn.io/v1/browser-callback`
  - Note the Callback URL should be `https://oauth.pstmn.io/v1/callback` with the "Authorize using browser" box checked if you are using the Postman Desktop app
- Auth URL: `https://www.linkedin.com/oauth/v2/authorization`
- Access Token URL: `https://www.linkedin.com/oauth/v2/accessToken`
- Client ID: {using the client_id from the environment variables}
- Client Secret: {using the client_secret from the environment variables}
- Scope: Differs per collection but an example is {`rw_ads,r_basicprofile,w_organization_social,w_member_social,rw_organization_admin`}
- Client Authentication: `Send client credentials in body` when the Grant Type is Authorization Code. `Send as Basic Auth header` when the Grant Type is Client Credentials.



Step 6 - Identity Authentication

If the Grant Type in Step 5 was Authorization Code then Postman will take you to the LinkedIn authorization page, where you may be prompted to log into LinkedIn. Click "Allow" to authorize the request. The prompt on the authorization page is dictated by the requested scopes in the previous step.



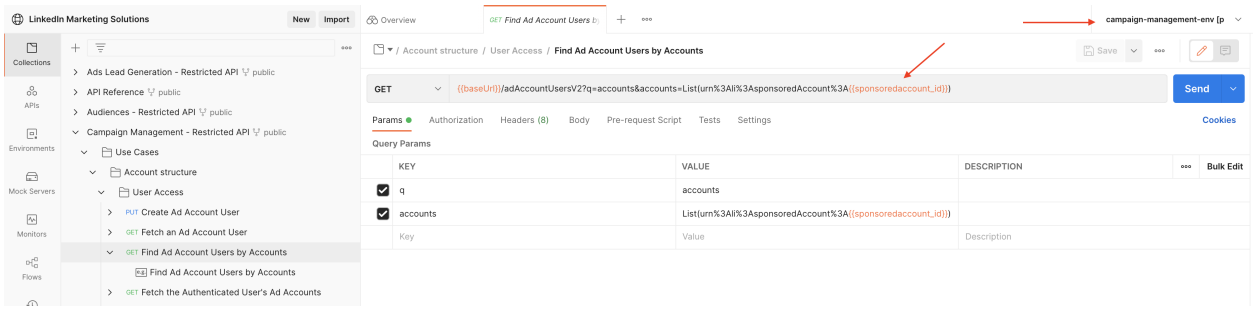
Step 7 - Use Token

Postman will then display your access token to be used for testing. Choose the 'Use Token' button to set this as the currently used token. The token will automatically be propagated to all requests within the corresponding collection. The video below shows an example of requesting a 3-legged token via the Authorization Code Grant Type.
https://www.microsoft.com/en-us/videoplayer/embed/RWQmh5?postJsllMsg=true

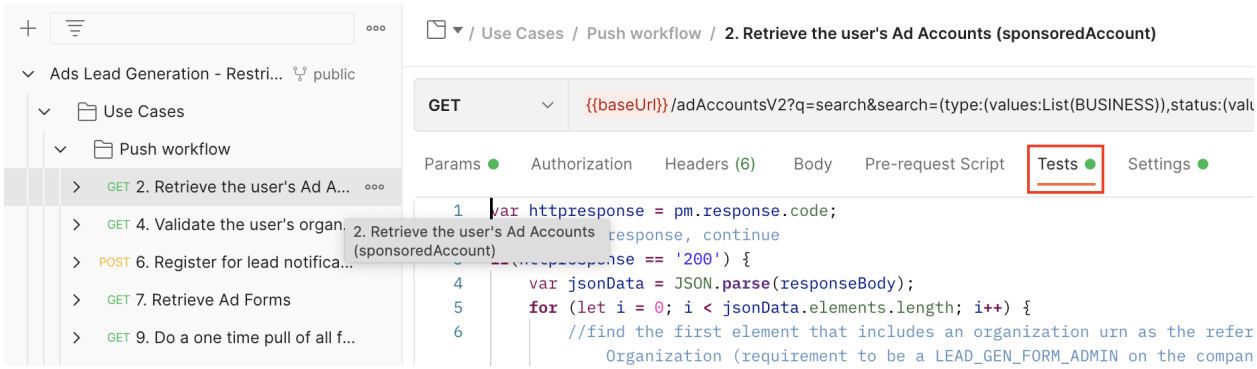
Step 8 - Testing

Finally, send a request within the Use Cases folder. Ensure the correct environment is selected and that if any environment or collection level variables are being used in the request, ensure they are set. For example, in the screenshot below, the request uses the `sponsoredaccount_id` variable from the `campaign-management-env` environment.

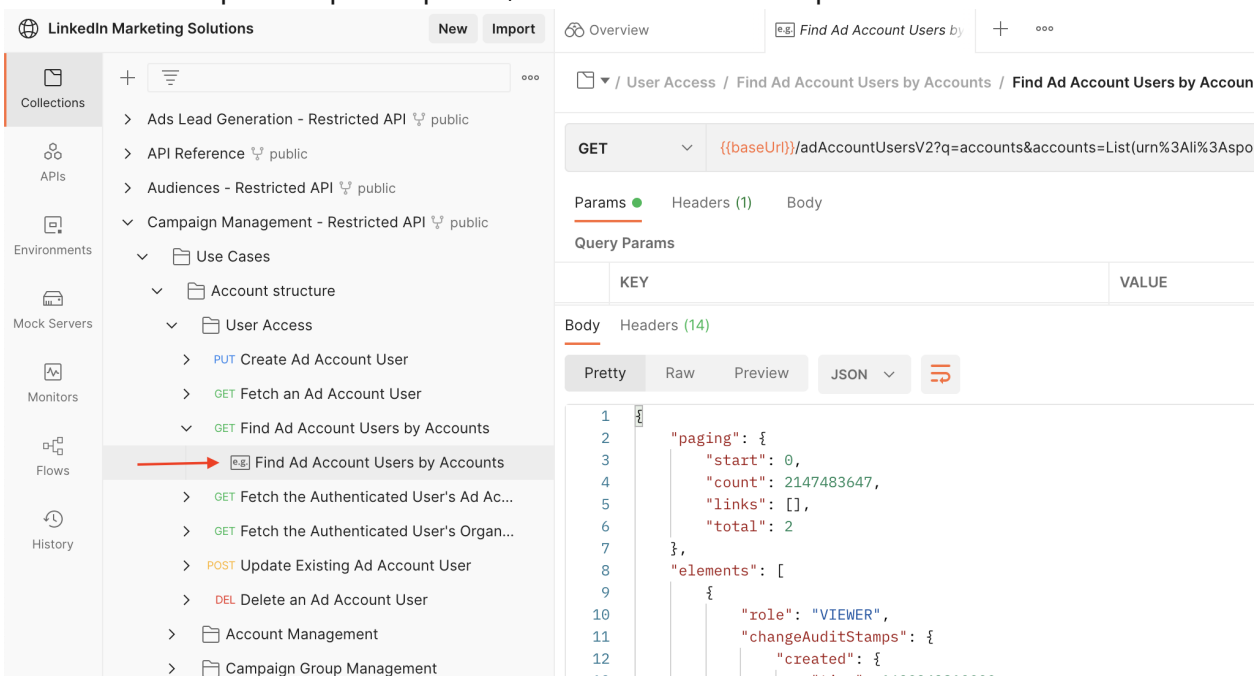
Learn more about Postman variables in Postman's online documentation



Note that some requests dynamically set variables via a script that runs post request execution. You will know if a script is set to run for a request if there is a green dot next to the Tests tab.



To see an example sample response, view the saved example.



Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Refresh Tokens with OAuth 2.0

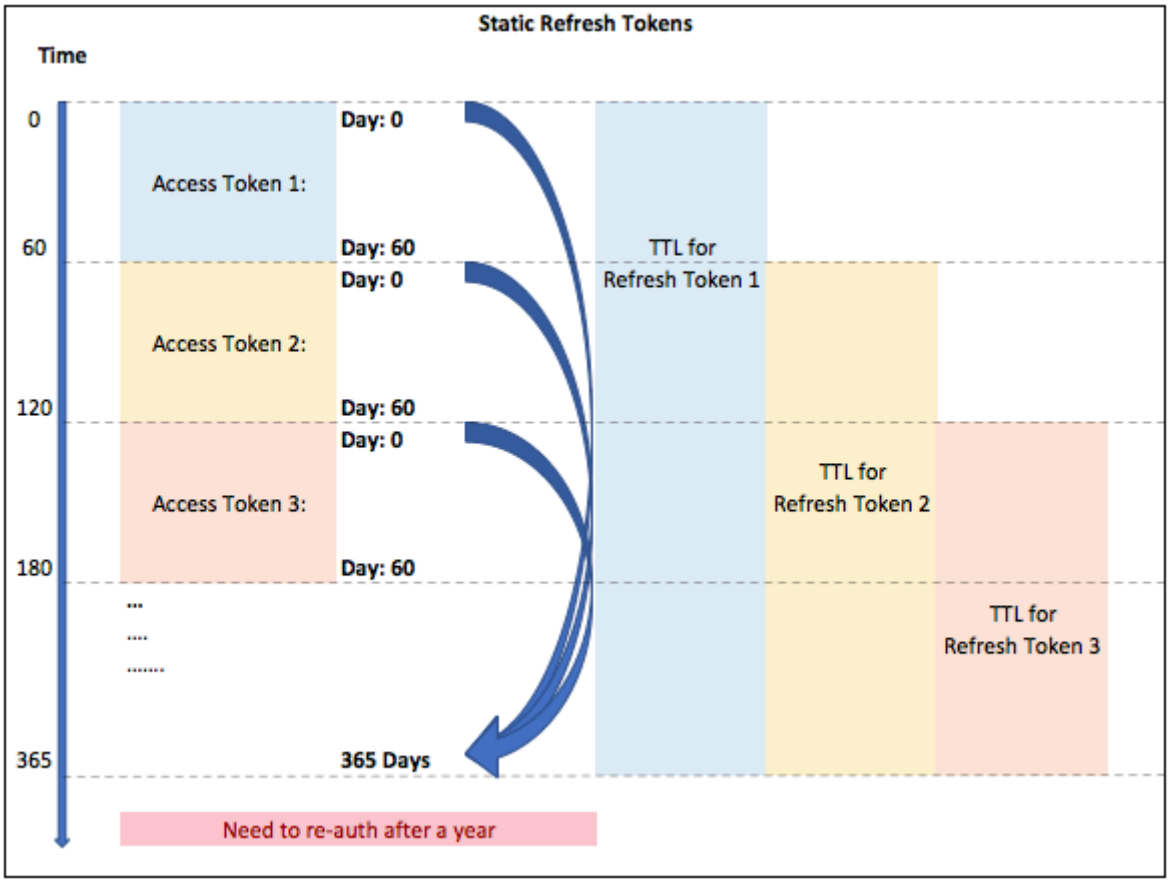
05/31/2025

LinkedIn supports programmatic refresh tokens for all approved Marketing Developer Platform (MDP) partners.

Introduction

Refresh tokens are used to get a new access token when your current access token expires. For more information, see the OAuth 2.0 RFC.

LinkedIn offers programmatic refresh tokens that are valid for a fixed length of time. By default, access tokens are valid for 60 days and programmatic refresh tokens are valid for a year. The member must reauthorize your application when refresh tokens expire.



When you use a refresh token to generate a new access token, the lifespan or Time To Live (TTL) of the refresh token remains the same as specified in the initial OAuth flow (365 days), and the new access token has a new TTL of 60 days.

For example, on:

- Day 1 - Your refresh token has a TTL of 365 days, and your access token has a TTL of 60 days.

- Day 59 - If you generate a new access token using the refresh token, the access token will have a TTL of 60 days and the refresh token will have a TTL of 306 days (365-59=306).
- Day 360- If you generate a new access token, your access token and refresh token will both expire in 5 days (365-360=5) and you must get your application reauthorized by the member using the authorization flow.

> ⓘ Note
>
> - Refresh Tokens are useful in minting new Access tokens and allow for seamless operations for extended periods of time. However, LinkedIn reserves the right to revoke Refresh Tokens or Access Tokens at any time due to technical or policy reasons. In such scenarios, the expectation from products leveraging Refresh Tokens is to fallback to the standard OAuth flow, and present the login screen to the end users.
> - To track the usage of refresh tokens, refer to the Developer Portal Tools page.

Step 1: Getting a Refresh Token

Use the Authorization Code Flow to get both a refresh token and access token. If your application is authorized for programmatic refresh tokens, the following fields are returned when you exchange the authorization code for an access token:

- `refresh_token` — Your refresh token for the application. This token must be kept secure.
- `refresh_token_expires_in` — The number of seconds remaining until the refresh token expires. Refresh tokens usually have a longer lifespan than access tokens.
- `scope` — URL-encoded, space-delimited list of member permissions your application has requested on behalf of the user.

Sample Response

JSON


```
{
  "access_token": "AQXNnd2kXITHELmWblJigbHEuoFdfRhOwGA0QNnumBI8XOVSs0HtOHEU-
wvaKrkMLfxxaB1O4poRg2svCWWgwhebQhqrETYlLikJJMgRAvH1ostjXd3DP3BtwzCGeTQ7K9vvAqfQK5i
G_eyS-q-
y8WNt2SnZKZumGaeUw_zKqtgCQavfEVCddKHcHLaLPGVUvjCH_KW0DJIdUMXd90kWqwuw3UKH27ki5raFD
PuMyQXLYxkqq4mYU-IUuZRwq1pcrYp1Vv-
ltbA_svUxGt_xeWeSxKkmgivY_DlT3jQylL44q36ybGBSbaFn-
UU7zzio4EmOzdmm2tlGwG7dDeivdPDsGbj5ig",
  "expires_in": 86400,
  "refresh_token": "AQWAft_WjYZKwuWXLC5hQlghgTam-tuT8CvFej9-
```

```
XxGyqeER_7jTr8HmjiGjqil13i7gMFjyDxh1g7C_G1gyTZmfcD0Bo2oEHofNAkr_76mSk84sppsGbygwW-
5oLsb_OH_EXADPIFo0kppznrK55VMIBv_d7SINunt-
7DtXCRAv0YnET5KroQOlmAhc1_HwW68EZniFw1YnB2dgDSxCkXnrfHYq7h63w0hjFXmgrdxeeAuOHBHnFF
YHOWWjI8sLLenPy_EBrgYIitXsAkLUGvZXlCjAWl-W459feNjHZ0SIsyTVwzAQtl5lmw1ht08z5Du-
RiQahQE0sv89eimHVg9VSNOaTvw",
  "refresh_token_expires_in": 525600,
  "scope":"r_basicprofile"

}
```


> **Note**
>
> Refresh tokens are approximately 500 characters long. We recommend that your application stack be made to handle tokens of at least 1000 characters to accommodate future expansion plans. This applies to access tokens as well as refresh tokens.

Step 2: Exchanging a Refresh Token for a New Access Token

You can exchange the refresh token for a new access token by making the following HTTP POST request with a `Content-Type` header of `x-www-form-urlencoded` and the following parameters in the request body:

POST

```
https://www.linkedin.com/oauth/v2/accessToken
```

Expand table

Parameter	Description	Required
grant_type	The value of this field should always be refresh_token.	Yes
refresh_token	The refresh token from Step 1.	Yes
client_id	The Client ID value generated when you registered your application.	Yes
client_secret	The Client Secret value generated when you registered your application.	Yes

Sample Request

https

```
POST https://www.linkedin.com/oauth/v2/accessToken
```

```
Content-Type: application/x-www-form-urlencoded
grant_type=refresh_token&refresh_token=AQQOMeCIQMa6-zjU-
02w8EJW67wPVk3hjJE5x1lZhU013LihKD8i1DpvaAl2jnuP8F1uXMgkm8nzjPfnaJR_kQNOxsLRLZWnAMz
HMm81S0yQlkBYicw&client_id=861hhm46p48to2&client_secret=gPecS7yqHkyyShvR
```

A successful request returns a new access token with a new expiration time and the refresh token.

JSON


```
{
  "access_token": "BBBB2kXITHELmWblJigbHEuoFdfRhOwGA0QNnumBI8XOVSs0HtOHEU-
wvaKrkMLfxxaB1O4poRg2svCWWgwhebQhqrETYlLikJJMgRAvH1ostjXd3DP3BtwzCGeTQ7K9vvAqfQK5i
G_eyS-q-
y8WNt2SnZKZumGaeUw_zKqtgCQavfEVCddKHcHLaLPGVUvjCH_KW0DJIdUMXd90kWqwuw3UKH27ki5raFD
PuMyQXLYxkqq4mYU-IUuZRwq1pcrYp1Vv-
ltbA_svUxGt_xeWeSxKkmgivY_DlT3jQylL44q36ybGBSbaFn-
UU7zzio4EmOzdmm2tlGwG7dDeivdPDsGbj5ig",
  "expires_in": 86400,
  "refresh_token": "AQWAft_WjYZKwuWXLC5hQlghgTam-tuT8CvFej9-
XxGyqeER_7jTr8HmjiGjqil13i7gMFjyDxh1g7C_G1gyTZmfcD0Bo2oEHofNAkr_76mSk84sppsGbygwW-
5oLsb_OH_EXADPIFo0kppznrK55VMIBv_d7SINunt-
7DtXCRAv0YnET5KroQOlmAhc1_HwW68EZniFw1YnB2dgDSxCkXnrfHYq7h63w0hjFXmgrdxeeAuOHBHnFF
YHOWWjI8sLenPy_EBrgYIitXsAkLUGvZXlCjAWl-W459feNjHZ0SIsyTVwzAQtl5lmw1ht08z5Du-
RiQahQE0sv89eimHVg9VSNOaTvw",
  "refresh_token_expires_in": 439200,
  "scope":"r_basicprofile"
}
```


API Error Details

Expand table

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
400	invalid_request "The provided authorization grant or refresh token is invalid, expired or revoked"	Invalid or expired or revoked refresh token is sent as part of the request.	Refresh Token expired or revoked or invalid, hence reauthenticate the member to generate the new refresh token.
400	invalid_request "A required parameter "redirect_uri" is missing"	Redirect_URI in the request is missing. It is mandatory parameter.	Pass the Redirect_URI in the request to route user back to correct landing page.
400	invalid_request "A required parameter "grant_type" is	Grant type in the request is missing. It is	Add grant_type as "refresh_token" in the request.

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
	missing"	mandatory parameter.	
400	invalid_request "A required parameter "client_id" is missing"	Client ID in the request is missing. It is mandatory parameter.	Pass the client id of the app in request.
400	invalid_request "A required parameter "refresh_token" is missing"	Refresh Token in the request is missing. It is mandatory parameter.	Pass the stored Refresh Token received as part of initial access token call.

Developer Portal Tools

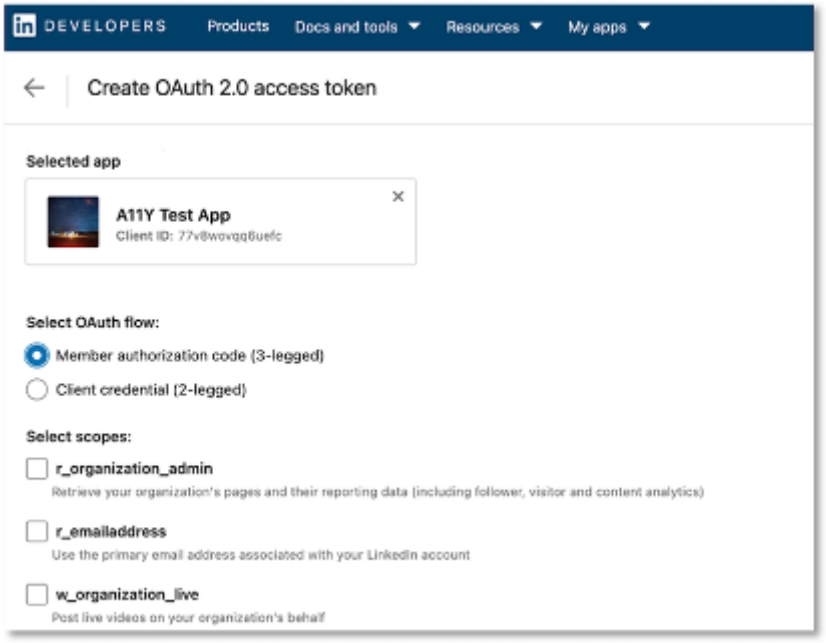
10/08/2025

The LinkedIn Developer Portal Token Generator Tool allows a quick and easy method for generating an access token to make authenticated API calls.

Generate a Token in the Developer Portal

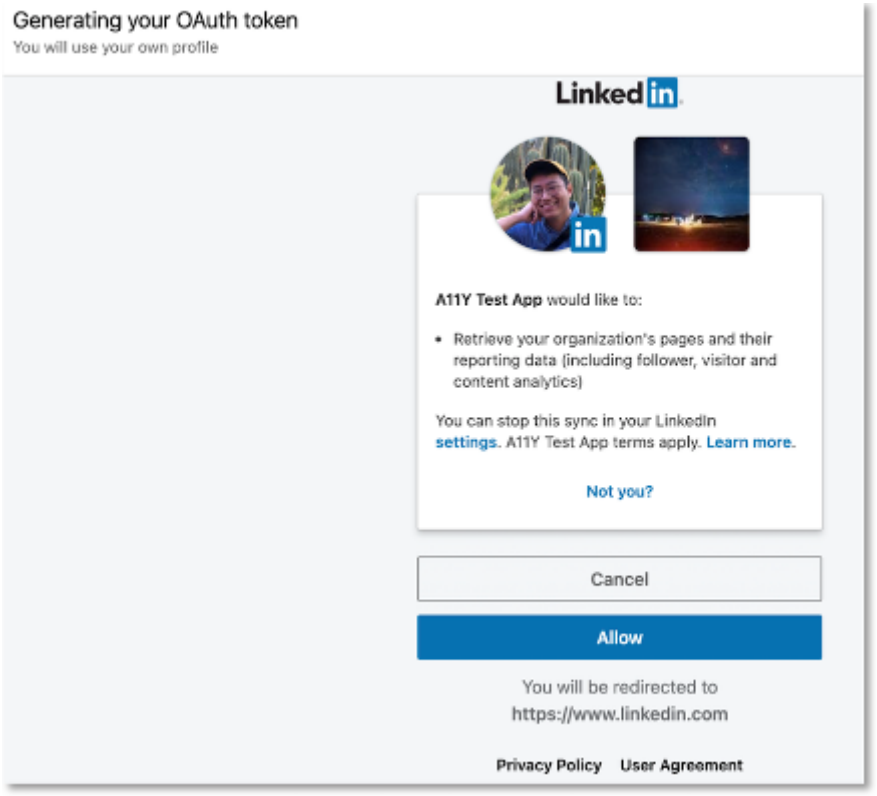
Once a token is generated, users are redirected to the token information page which includes details like OAuth scopes and token time to live (TTL) for reference during development activities.

1. Visit the LinkedIn Developer Portal Token Generator tool.
2. Select the app you'd like to generate a token for.
3. Select OAuth flow and permission scopes.



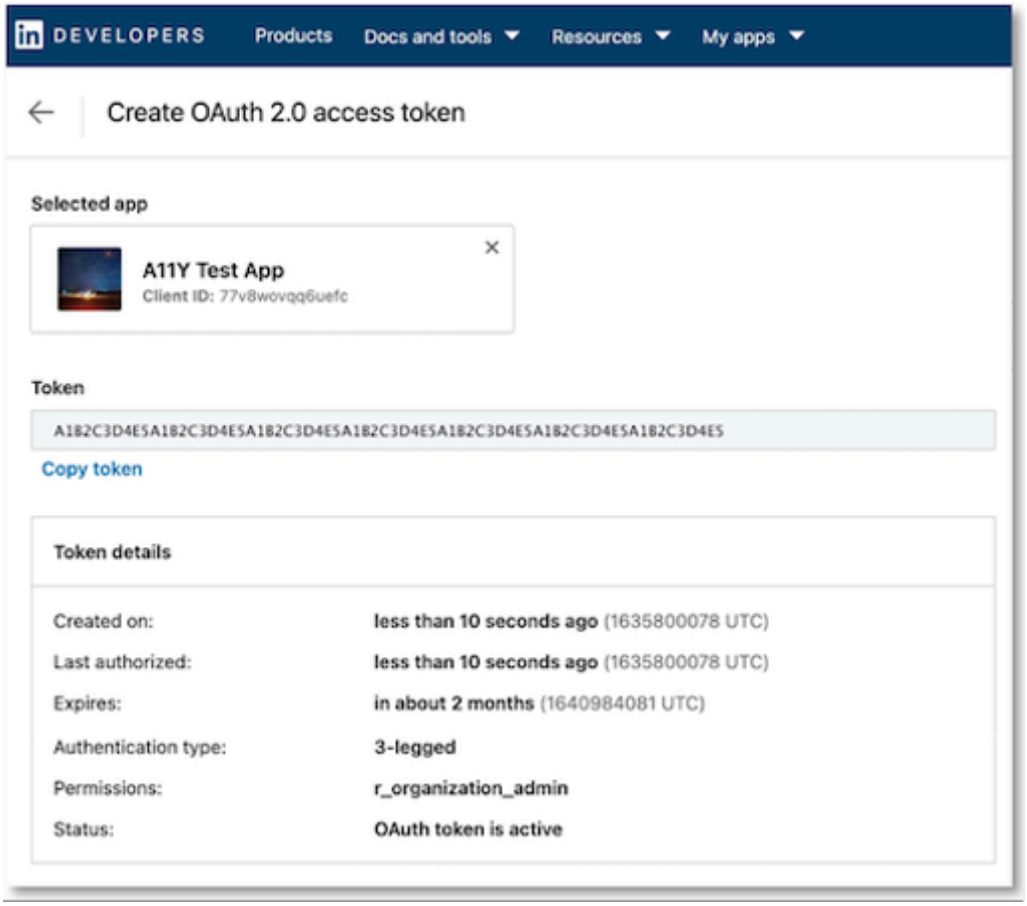
4. Member approval

The authenticated member will receive a request for your app to access to their profile.

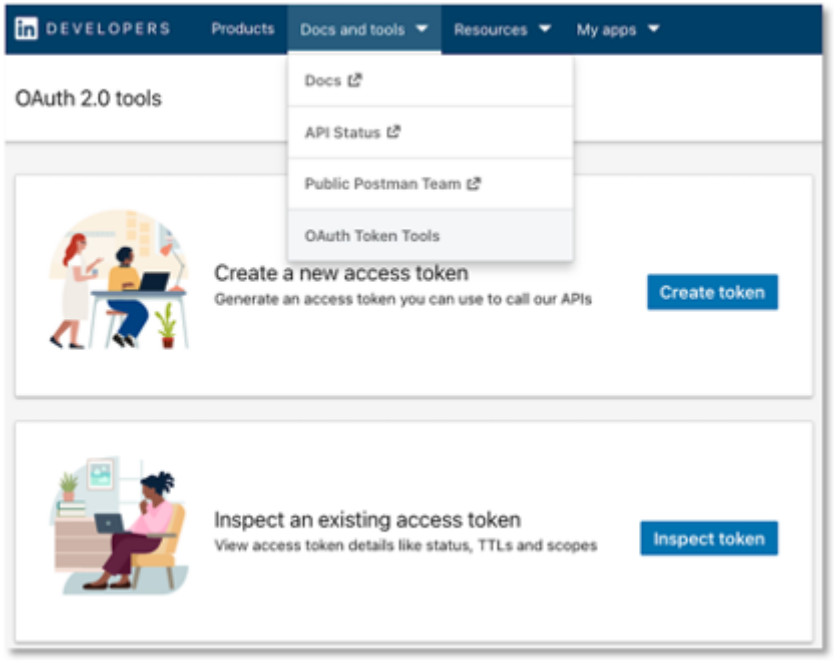


5. Token Generation

Once the token is generated, the "Token Details" will be shown along with the token. Click "Copy token" to paste it into your application code.



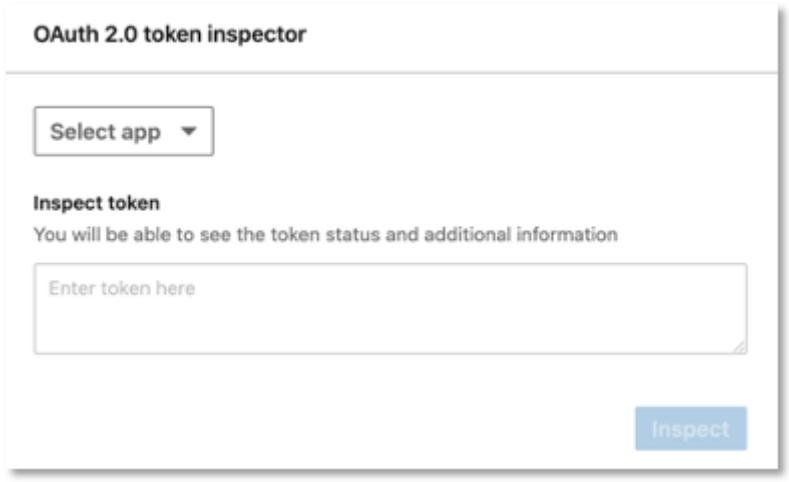
Should you wish to verify this an existing token, the Token Inspector tool is available on the same page as the token generator.



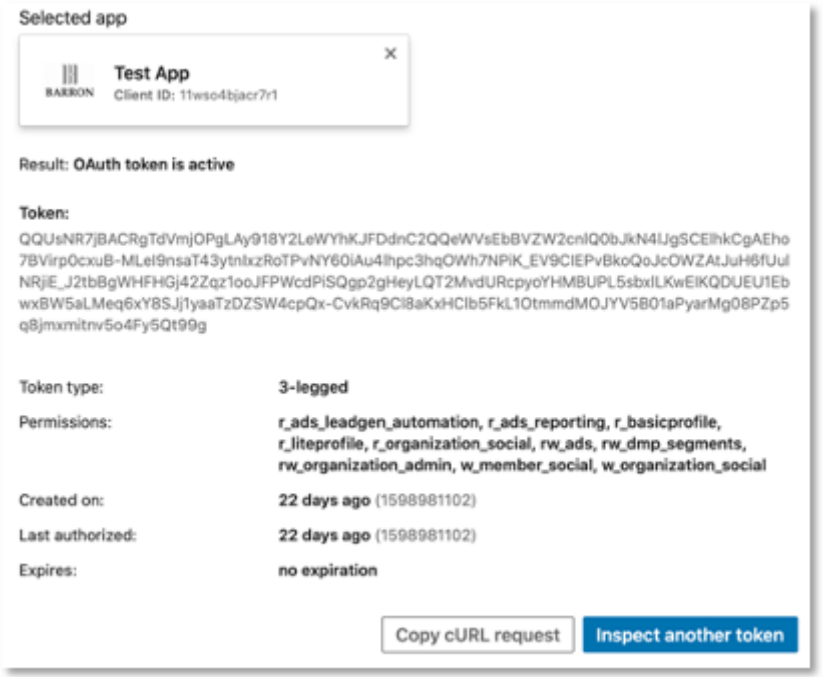
Developer Portal Token Inspector

LinkedIn's Developer Portal has a token inspector tool to make token validation as simple as copy and paste. The same Token validation is available through the API or the UI. The OAuth 2.0 token inspector is accessible from the developer portal under "Docs and Tools" in the navigation bar.

The tool requires you to select a developer application either from a dropdown or by entering the client ID if you have more than 10 developer applications. Make sure you have created at least one developer application or have been added as an Admin team member to a developer application before using the tool. You may only inspect tokens generated by the selected developer application.



The tool can also be used to generate a curl request to the token introspection endpoint. Simply paste a token in the text box, click "Inspect", and use the "Copy cUrl request" button.



OAuth Refresh Token Usage

A table has been added to the bottom of the Analytics Dashboard in the Developer Portal to track OAuth refresh token usage. It shows the percentage of quota used for generating access tokens via refresh tokens, the total allowed quota per app per day, and whether the app is currently Throttled or Not Throttled. The quota resets daily at 00:00 UTC, and a countdown timer indicates the time left until the next reset.

Token Introspection

Article • 02/05/2025

Introduction

The token inspector tool enables developers to check the Time to Live (TTL) and status (active/expired) for all tokens (including Enterprise tokens.) For Authorization Code Flow (3-legged OAuth) tokens, permission scopes will be displayed. You can fetch access token data using the `/introspectToken` endpoint or the Token Inspector Tool in the UI.

API Details

https

```
POST https://www.linkedin.com/oauth/v2/introspectToken

Content-Type: application/x-www-form-urlencoded
```

Sample Request

http

HTTP

```
POST https://www.linkedin.com/oauth/v2/introspectToken
```

Request Body

Expand table

Field	Type	Description
client_id	string	Required. Application client id
client_secret	string	Required. Application client secret
token	string	Required. The string value of the token returned using Client Credential Flow (2-legged OAuth), Authorization Code Flow (3-legged OAuth), or `Enterprise_User` (Enterprise OAuth Flow).

Sample Response

JSON

```
{
    "active": true,
    "client_id": "xxxxxxxx",
    "authorized_at": 1493055596,
    "created_at": 1493055596,
    "status": "active",
    "expires_at": 1497497620,
    "scope": "r_liteprofile,r_emailaddress,w_member_social",
    "auth_type": "_see note below_"
}
```

> **Note**
>
> Possible `auth-type` values returned are:
>
> "auth_type": "2L"
>
> "auth_type": "3L"
>
> "auth_type": "Enterprise_User"

Response Fields

Expand table

Field	Type	Description
active	boolean	Required. Boolean indicator of whether or not the returned token is currently active
status	string	Optional. An enum string with values: `revoked` - Token has been revoked `expired` - Token has expired due to the "expires_at" TTL `active` - Token is active
scope	string	Optional. A string containing a comma-separated list of scopes associated with this token. Returned only for token obtained via Authorization Code Flow (3-legged OAuth)
client_id	string	Optional. Optional. Application Client ID
created_at	long	Optional. Epoch time in seconds, indicating when this token was originally issued

Field	Type	Description
expires_at	long	Optional. Epoch time in seconds, indicating when this token will expire
authorized_at	long	Optional. Epoch time in seconds, indicating when the token was authorized
auth_type	string	Optional. String with values: `3L` - 3-legged member token `2L` - 2-legged application token `Enterprise_User` - Enterprise member token

HTTP Response Status Codes

The response will vary depending on the status of the token and its authenticity.

Expand table

Status Code	Description
200	Success
400	Invalid `client id` or `token`
401	Invalid `client secret`

> **Note**
>
> If the credentials are valid but do not match the client information in the token, you will receive a successful response (status 200 OK), however with `"active": false,` in the response body.

Feedback

Was this page helpful? Yes No

Provide product feedback

Application Secret Management

Manage secrets for your parent, child, and standalone applications to keep your integrations secure. You can rotate secrets on demand or on a regular schedule, and manage up to two secrets per application. You can perform the following actions to manage secrets:

Rotate a Secret

Rotate your application secrets regularly to maintain security. When you rotate a secret, the new secret gets added without removing the old one—both remain active so you can update your integration without downtime.

> ⓘ **Important**
>
> - Each application can have a maximum of two secrets
> - If two secrets already exist, delete one before rotating, else the rotation API request will fail

Rotate Parent or Standalone Application Secret

Endpoint

HTTP

```
POST https://api.linkedin.com/v2/developerApplicationsSecurity?
action=rollDeveloperApplicationSecret
```

Headers

Expand table

Header Name	Value
X-RestLi-Method	action
X-RestLi-Protocol-Version	2.0.0
Authorization	Bearer 2L/3L token

> **Note**
>
> Generate the Bearer token using the application's credentials.

Request Body

JSON

```
{}
```

Request Body Fields

Expand table

Field	Description	Format	Required
{}	Empty braces	Object	Yes

Response

Success: HTTP 200 OK

Response Body:

JSON

```
{
  "value": {
    "client_secret": "bFWEECAwQp1AT6rJ"
  }
}
```

Rotate Child Application Secret

Endpoint

HTTP

```
POST https://api.linkedin.com/v2/developerApplicationsSecurity?
action=rollDeveloperApplicationSecret
```

Headers

Expand table

Header	Value
X-RestLi-Method	action
X-RestLi-Protocol-Version	2.0.0
Authorization	Bearer 2L/3L `<token>`

> **Note**
>
> - Use your parent application credentials to generate the Bearer token for child applications.

Request Body

JSON

```
{
  "childDeveloperApplication": "urn:li:developerApplication:<child-application-id>"
}
```

Request Body Fields

Expand table

Field	Description	Type	Required
childDeveloperApplication	The URN of the child application (for example, `urn:li:developerApplication:123456`).	String	Yes

Sample Request (Shell)

Bash

```
curl --location "https://api.linkedin.com/v2/developerApplicationsSecurity?
action=rollDeveloperApplicationSecret" \
-X POST \
-H 'X-RestLi-Method:action' \
-H 'Accept:application/json' \
-H 'Content-Type:application/json' \
-H 'X-RestLi-Protocol-Version:2.0.0' \
-H 'Authorization: Bearer <token>' \
--data '{
  "childDeveloperApplication": "urn:li:developerApplication:123456"
}'
```

Sample Response

Success: HTTP 200 OK

JSON

```
{
  "value": {
    "client_secret": "bFWEECAwQp1AT6rJ"
  }
}
```

Delete a Secret

Delete a secret when you need to rotate but already have two active secrets, or when removing a secret that's no longer needed. While deleting a secret, the same secret needs to be passed as part of the API payload.

Delete Parent or Standalone Application Secret

Endpoint

HTTP

```
POST https://api.linkedin.com/v2/developerApplicationsSecurity?
action=removeDeveloperApplicationSecret
```

Request Headers

Expand table

Header Name	Value
X-RestLi-Method	action
X-RestLi-Protocol-Version	2.0.0
Authorization	Bearer 2L/3L token

Request Body

JSON

```
{
  "secret": "<your_app_secret>"
}
```

Request Body Fields

Expand table

Field	Description	Type	Required
secret	The secret to be deleted	String	Yes

Sample Request (Shell)

Bash

```
curl --location "https://api.linkedin.com/v2/developerApplicationsSecurity?
action=removeDeveloperApplicationSecret" \
-X POST \
-H 'X-RestLi-Method:action' \
-H 'Accept:application/json' \
-H 'Content-Type:application/json' \
-H 'X-RestLi-Protocol-Version:2.0.0' \
-H 'Authorization: Bearer <2L/3L token>' \
--data '{
  "secret": "<your_app_secret>"
}'
```

Response

Success: HTTP 200 OK

The API returns an HTTP 200 status code with an empty response body when the secret is successfully deleted.

Delete Child Application Secret

Endpoint

HTTP

```
POST https://api.linkedin.com/v2/developerApplicationsSecurity?
action=removeDeveloperApplicationSecret
```

Request Headers

Expand table

Header	Value
X-RestLi-Method	action
X-RestLi-Protocol-Version	2.0.0
Authorization	Bearer 2L/3L `<token>`

Request Body

JSON

```
{
  "childDeveloperApplication": "urn:li:developerApplication:<child-application-id>",
  "secret": "<your_app_secret>"
}
```

Request Body Fields

Expand table

Field	Description	Type	Required
childDeveloperApplication	The URN of the child application.	String	Yes
secret	The secret to be deleted.	String	Yes

Response

Success: HTTP 200 OK

The API returns an HTTP 200 status code with an empty response body when the secret is successfully deleted.

Sample Request (Shell)

Bash

```
curl --location "https://api.linkedin.com/v2/developerApplicationsSecurity?
action=removeDeveloperApplicationSecret" \
-X POST \
-H 'X-RestLi-Method:action' \
-H 'Accept:application/json' \
-H 'Content-Type:application/json' \
-H 'X-RestLi-Protocol-Version:2.0.0' \
-H 'Authorization: Bearer <token>' \
--data '{
  "childDeveloperApplication": "urn:li:developerApplication:123456",
  "secret": "<your_app_secret>"
}'
```

Error Codes

Expand table

HTTP code	Status	Description
500	Internal Server Error	• Occurs when a different parent-app-id or secret token is used for a child-app not linked to the selected parent-app. • Removing the last remaining secret from a child-app. • Attempting to create a third secret when two already exist.

> **Note**
>
> These error codes are specific to secret rotation and deletion APIs. Error codes may change in future updates.

Last updated on 11/05/2025

Concepts

Article • 05/08/2023

LinkedIn's v2 REST APIs contain a number of concepts that help apply standard RESTful principles at scale, model data consistently, and ultimately provide a well-structured end-to-end developer experience.

https

```
host: api.linkedin.com
basePath: /v2
scheme: https
```

> **Note**
>
> The `/v2` basePath is used for *most* LinkedIn API calls.
>
> The ***authentication procedure*** uses host: `www.linkedin.com`

- Data Formats
- Decoration
- Errors
- Methods
- Pagination
- Projections
- Protocol Version
- Rate Limits
- URNs

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Data Formats

Article • 05/08/2023

Input

When you create or update existing entities, specify all input in JSON format and use the following HTTP Content Type:

```
Content-Type: application/json
```

XML input is not supported.

Output

The output is returned in JSON format as the following HTTP Content Type:

```
Content-Type: application/json
```

XML output is not supported.

Feedback

Error Handling

Article • 03/24/2025

HTTP requests can fail for a variety of reasons. LinkedIn provides standard HTTP status codes and clear and concise messages to help you easily understand these errors.

Sample Response

JSON

```
{
    "message": "Empty oauth2_access_token",
    "serviceErrorCode": 401,
    "status": 401
}
```

Error responses have the following details:

- `message` - A description of the error.
- `serviceErrorCode` - A subcode that further classifies the error.
- `status` - The type of error (status code).

HTTP Status Codes and Error Types

LinkedIn uses standard HTTP status codes for each API's response.

Status codes are divided into the following five categories:

- 1xx: Informational - Communicates transfer protocol-level information.
- 2xx: Success - The request was successful.
- 3xx: Redirection - The client must take some additional action to complete the request.
- 4xx: Client Error - Failed request due to client error.
- 5xx: Server Error - Failed request due to server error.

400 Bad Request

A `400 Bad Request` error means that the server was unable to proceed with the request. The most common cause of the error is bad syntax in the request URL or body. To fix a `400 Bad Request` error, do the following:

- Check if an invalid character is included in the URL.
- Check API examples.
- Ask your colleagues for help.
- Talk to the rubber duck.

401 Unauthorized

`401 Unauthorized` errors are usually caused by a problem in the request header of your API call. For example, if you don't use a valid access token when you make an API call on behalf of a LinkedIn member, a `401 Unauthorized` error is returned. Some common cases are:

Expand table

Error Type	Fix
Unknown authentication schema	Unrecognized authentication header schema. Make sure the authentication header follows the format `Authorization: Bearer (your access token)`
Empty OAuth2 access token	The authentication header is missing or empty. Make sure the authentication header follows the format `Authorization: Bearer (your access token)`
Invalid access token	Incorrect access token, make sure you follow the authentication procedure to get a correct access token.
Expired access token	The access token has expired, see how to refresh your access token
The token has been revoked	The access token has been revoked by the member from their privacy settings on LinkedIn's website. To continue using your application, the member has to re-authenticate to get a new access token for your application.

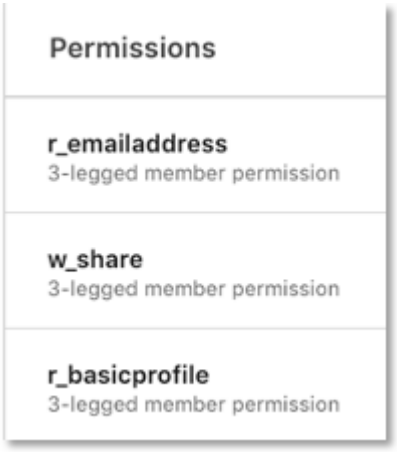
> **Note**
>
> Access token downstream verification failures return a 500 Internal Server Error.

403 Access Denied

When your application makes an API call with a member's access token, LinkedIn checks if the access token has permission to access the API. If the access token does not have

the correct permissions, a `403 Access Denied` error is returned. When this happens, check the following:

- Does your application have permissions to make the API call?
- Does your application ask the user to enable these permissions?
- Did your application change the scope while requesting an access token?



If you continue to see the error, reach out to your partner technical support channel or https://developer.linkedin.com/support.

404 Resource Not Found

This error occurs when your application tries to call an API or fetch an entity that does not exist. For example, the API to get a friend's profile is `/v2/people/id={personId}`, not `/v2/person/id={personId}`. In some cases (Ads, for example), a 404 error is returned when attempting to access a restricted API. See 403 Access Denied and contact your partner technical support channel if you continue to see the error.

405 Method Not Allowed

This error indicates that the HTTP protocol methods in your request are not supported. Check the documentation for the API to see supported methods.

411 Length Required

This error indicates that the server refuses to accept the request without a defined `Content-Length` header. Please make sure POST requests with an empty body have a `Content-Length` header specified.

426 Version Header is Deprecated

This error indicates that the version passed in the request header `X-LinkedIn-Version` has been deprecated (e.g., 202401). This occurs because all versions are scheduled for deprecation after a specific time window (e.g., 12 months). Refer to the Sample Response for more information.

429 Rate Limit

On LinkedIn's platform, all API requests that you make are rate limited to prevent abuse and to ensure service stability. These errors will return an error message of "Resource level throttle limit for calls to this resource is reached." If you get a `429 Rate Limit` error, check if too many redundant calls are being made and review your application's Usage & Limits in the Developer Portal. If you've confirmed that the current rate limits do not meet your application's needs, contact us if you are our partner, or join our partner program through https://developer.linkedin.com/partner-programs.

In rare cases, LinkedIn may also return a 429 response as part of infrastructure protection. API service will return to normal automatically.

500 Internal Server Error

A `500 Internal Server Error` indicates that LinkedIn is experiencing an internal error. If you continue to receive server errors, record the following details and report it to your partner technical support channel or https://developer.linkedin.com/support:

- Request: `url`, `method`, `header`. For example, `access_token`, `body`.
- Response: `header`. For example, `x-li-uuid`, `x-li-fabric`, `x-li-request-id`, `body`.
- Your application configuration. For example, Client ID.

504 Gateway Timeout

A `504 Gateway Timeout` error happens when it takes too long for LinkedIn to process your API call. Due to the nature of cloud APIs, LinkedIn's services may be occasionally interrupted or temporarily unavailable for reasons outside of its control. Make sure you have proper error handling logic, such as caching and retry patterns, to cover these issues. If your application continues to receive these errors, contact https://developer.linkedin.com/support to report the issue.

Feedback

Was this page helpful? Yes No

Provide product feedback

Field Projections

Article • 05/08/2023

Field projection controls how much of an entity's data is displayed in response to an API request.

All APIs have a default set of field projections that control which fields are returned. If you don't need certain fields, you can decrease response time and payload size by using a projection to ask only for the fields your application is interested in.

By default, services may not always return all of the information that your application requests. In these cases, you'll need to use a projection to retrieve any non-default fields.

Field projections are defined using the `&fields=` query parameter and narrowed by providing a comma-separated list of field names that you want returned as the value of the parameter.

In the following example, a service returns these types of objects:

Sample Objects

JSON

```
{
    "id" : int,      <- Default projection field
    "foo": string,   <- Default projection field
    "bar": boolean,
    "baz": Object
}
```

A GET call to retrieve one of these objects provides the following response:

https

```
GET https://api.linkedin.com/v2/sampleService/42
```

Sample Response

JSON

```
{
    "foo": "Zing!",
```

```
    "id": 42
}
```

Notice how the `bar` and `baz` fields were not returned in the response. This is because they are not part of the service's default field projection.

If you want to get `id`, `bar`, and `baz` back in the response (but not `foo`, because it's irrelevant to your application), use a field projection:

https

```
GET https://api.linkedin.com/v2/sampleService/42?fields=id,bar,baz
```

Sample Response

JSON

```
{
    "bar": true,
    "baz": {
        "beep": "Yay!",
        "bloop": "Meh",
        "blorp": "Boo!"
    },
    "id": 42
}
```

Child Objects

In the above example, `baz` is returned in the response due to the field projection specified because `baz` is an object that has its own fields. Use field projections to select the data you need.

Use the `parentField:(child**Field_1,…,childField_n**)` syntax to select fields for a child object:

https

```
GET https://api.linkedin.com/v2/sampleService/42?fields=id,baz:(beep)
```

Sample Response

JSON

```
{
    "baz": {
        "beep": "Yay!"
    },
    "id": 42
}
```

Parent Field as Map

Objects can have nested objects with child fields of their own such as the example below:

Sample Data

JSON

```
{
    "baz": {
        "1": {
            "beep": "Yay!",
            "foo": "foo1"
        },
        "2": {
            "beep": "Nay!",
            "foo": "foo2"
        }
    },
    "id": 42
}
```

Use the `$*:(childField_1,…,childField_n)` syntax to control the fields requested from nested objects:

https

```
GET https://api.linkedin.com/v2/sampleService/42?fields=id,baz:($*:(beep))
```

Sample Response

JSON

```
{
    "baz": {
```

```
        "1": {
            "beep": "Yay!"
        },
        "2": {
            "beep": "Nay!"
        }
    },
    "id": 42
}
```

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Pagination

Article • 05/08/2023

API calls that return a large number of entities are broken up into multiple pages of results. You might need to make multiple API calls with slightly varied paging parameters to iteratively collect all the data you are trying to gather.

Use the following query parameters to paginate through results:

Parameters

Name	Description	Default
start	The index of the first item you want results for.	0
count	The number of items you want included on each page of results. There could be fewer items remaining than the value you specify.	10

To paginate through results, begin with a `start` value of 0 and a `count` value of N. To get the next page, set `start` value to N, while the `count` value stays the same. Subsequent pages start at 2N, 3N, 4N, and so on.

Samples

Sample Request

https

```
GET https://api.linkedin.com/v2/{service}
```

Sample Response

JSON

```
"elements": [
    {"Result #0"},
    {"Result #1"},
    {"Result #2"},
    {"Result #3"},
    {"Result #4"},
    {"Result #5"},
```

```
    {"Result #6"},
    {"Result #7"},
    {"Result #8"},
    {"Result #9"}
],
"paging": {
    "count": 10,
    "start": 0
}
```

Sample Request

https

```
GET https://api.linkedin.com/v2/{service}?start=10&count=10
```

Sample Response

JSON

```
"elements": [
    {"Result #10"},
    {"Result #11"},
    {"Result #12"},
    {"Result #13"},
    {"Result #14"},
    {"Result #15"},
    {"Result #16"},
    {"Result #17"},
    {"Result #18"},
    {"Result #19"}
],
"paging": {
    "count": 10,
    "start": 10
}
```

End of the Dataset

You have reached the end of the dataset when your response contains fewer elements in the `entities` block of the response than your `count` parameter request.

Feedback

Protocol Versions

Article • 05/08/2023

LinkedIn V2 APIs support two protocol versions: 1.0 and 2.0. See the following syntax differences to help you migrate from 1.0 to 2.0.

LinkedIn plans to deprecate protocol version 1.0 in the near future. We strongly encourage developers to migrate as soon as possible and take advantage of the performance improvements in protocol version 2.0.

To use version 2.0, you must pass `X-Restli-Protocol-Version: 2.0.0` as the header in your API requests. If you don't pass a header, your call will default to version 1.0.

Single Resource Key

When performing a GET, PARTIAL_UPDATE, or DELETE on a resource, you must supply the resource key. See the following protocol version 1.0 example:

```
https

GET https://api.linkedin.com/urn:li:endorsement:(urn:li:person:2qXA98-
mVk,65761962366)
```

In protocol version 2.0, you must URL encode the entire resource key. The `(` must be encoded to `%28`, `)` to `%29`, `:` to `%3A` and `,` to `%2C`. See the following example:

```
https

GET
https://api.linkedin.com/v2/endorsement/urn%3Ali%3Aendorsement%3A%28urn%3Ali
%3Aperson%3A2qXA98-mVk%2C65761962366%29
```

Note that special characters in a params string not part of a resource key should not be encoded. See the following example where parentheses are not encoded:

```
https

GET https://api.linkedin.com/v2/ugcPosts?
q=authors&authors=List(urn%3Ali%3Aorganization%3A12345)
```

In cases where only numeric ID are passed in as the primary resource key, nothing will change from 1.0 to 2.0. Some APIs require a compound or complex key. The following

table lists the syntax differences:

Key Type	Version 1.0	Version 2.0
primitive key, long	3	3
primitive key, string	someString	someString
compound key with association	stringKey=string&longKey=5	(stringKey:string,longKey:5)
complex key	$param.a=value&$param.b=value &keyPart1=value1&keyPart2=value2	($param:(a:value,b:value), keyPart1:value1,keyPart2=value2)

Multiple Resource Keys

To perform a `BATCH_GET` in protocol version 1.0:

```
https

GET https://api.linkedin.com/people?ids=1&ids=2&ids=3
```

In protocol version 2.0, you must change this query to a `List` syntax:

```
https

GET https://api.linkedin.com/v2/people?ids=List(1,2,3,4)
```

Complex Keys

In protocol version 1.0, if a resource has complex keys:

```
https

GET https://api.linkedin.com/people?
ids[0].$params.a=value0A&ids[0].$params.b=value0B&ids[1].$params.a=value1A&i
ids[1].$params.b=value1B&ids[2].$params.a=value2A&ids[2].$params.b=value2B
```

In protocol version 2.0, if a resource has complex keys:

```
https
```

```
GET https://api.linkedin.com/people?
ids=List(($params.a:value0A,$params.b:value0B),
($params.a:value1A,$params.b:value1B),($params.a:value2A,$params.b:value2B))
```

> ⓘ Note
>
> The values and the resource parameters must be URL encoded but not the `,` grouping the fields and values. The single resource key had the `,` encoded because it was part of the whole value.

Parameters

When performing a `FINDER` in version 1.0, you often have query parameters such as filters represented as an array:

https

```
GET https://api.linkedin.com/resource?
q=FinderParam&param.aList[0]=foo&param.aList[1]=bar&param.aList[2]=baz&
param.anObject.aField=1&param.anObject.anotherField=value
```

In protocol version 2.0, you must change this query to a `List` syntax, much like the multiple keys:

https

```
GET https://api.linkedin.com/resource?q=myFinder&param=
(aList:List(foo,bar,baz),anObject:(aField:1,anotherField:value))
```

> ⓘ Note
>
> The values and the resource parameters must be URL encoded but not the `,` grouping the fields and values. The single resource key had the `,` encoded because it was part of the whole value.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Query Tunneling

Article • 05/08/2023

To improve our network infrastructure and better serve API traffic globally, LinkedIn will begin rejecting API calls not meeting new criteria.

> ⓘ **Note**
>
> For more information about API specific examples, see our ***Migration Guide***.

Requirements

Ensure your LinkedIn API requests comply with the following size requirements. If your request exceeds any of the size requirements listed below, your request will receive a 414 response.

Request parameter	**Size in KB**
Raw URL	8 KB max length (scheme + hostname + port + path + query string of the URL)
Query String	4 KB max length
Request URL	28 KB max length (headers + cookies + URI + queryString, but excluding POST data)
URL path segment	4 KB max characters (the area between slashes in a URL)

Query Tunneling

To support these requirements, we're introducing the concept of query tunneling. This feature allows you to modify your existing requests with a custom header to easily resolve offending requests.

Requests without Body

1. Change the request from `GET` to `POST`.
2. Add the `X-HTTP-Method-Override` header, using the original HTTP method (-H "X-HTTP-Method-Override: GET").

3. Add the `Content-Type` header (-H "Content-Type: application/x-www-form-urlencoded").
4. Move the query string to the body of the request.

Example

Let's use the example request below to see how we can quickly convert our existing request to the organizationalEntityAcls API using query tunneling.

Current Request

curl

```
curl -X GET 'https://api.linkedin.com/v2/organizationalEntityAcls?
q=roleAssignee&projection=(elements*(organizationalTarget~))' \
-H 'Authorization: Bearer redacted'
```

1. Change the request type from GET to POST.

curl

```
curl -X POST 'https://api.linkedin.com/v2/organizationalEntityAcls?
q=roleAssignee&projection=(elements*(organizationalTarget~))' \
-H 'Authorization: Bearer redacted'
```

2. Add the method override header and append the original GET request type.

curl

```
curl -X POST 'https://api.linkedin.com/v2/organizationalEntityAcls?
q=roleAssignee&projection=(elements*(organizationalTarget~))' \
-H 'X-HTTP-Method-Override: GET'
-H 'Authorization: Bearer redacted'
```

3. Add the `Content-Type` header.

curl

```
curl -X POST 'https://api.linkedin.com/v2/organizationalEntityAcls?
q=roleAssignee&projection=(elements*(organizationalTarget~))' \
-H 'X-HTTP-Method-Override: GET'
-H 'Content-Type: application/x-www-form-urlencoded'
-H 'Authorization: Bearer redacted'
```

4. Move the query string of the request URL to the request body.

curl

```
curl -X POST 'https://api.linkedin.com/v2/organizationalEntityAcls' \
-H 'X-HTTP-Method-Override: GET'
-H 'Content-Type: application/x-www-form-urlencoded'
-H 'Authorization: Bearer redacted'
--data 'q=roleAssignee&projection=(elements*(organizationalTarget~))'
```

Requests with Body

1. Change the request type from PUT to POST if the original request type was PUT. If the original request type was POST, keep it as POST.
2. Add the X-HTTP-Method-Override header using the original HTTP method `-H 'X-HTTP-Method-Override: POST'` for POSTs or `-H 'X-HTTP-Method-Override: PUT'` for PUTs.
3. Add the Content-Type header `-H 'Content-Type: multipart/mixed; boundary=xyz'`. Note that here we need to specify a boundary delimiter (here we use `xyz` for illustration) for multipart body, this delimiter needs to be unique and not appearing in your request content body or url.
4. Move the query string and original request body to body as two sections explained above.

Example

Let's use the example request below to see how we can quickly convert our existing request to the adCreativesV2 API using query tunneling.

Current Request

curl

```
curl -X POST 'https://api.linkedin.com/v2/adCreativesV2?ids=List(47770196)'
\
-H 'Authorization: Bearer redacted' \
-H 'Content-Type: application/json' \
-H 'X-Restli-Protocol-Version: 2.0.0' \
-H 'X-RestLi-Method:  BATCH_PARTIAL_UPDATE' \
--data '{"entities": {"47770196": {"patch": {"$set": {"status":
"ACTIVE"}}}}}'
```

1. Change the request type from PUT to POST if the original request type was PUT. If the original request type was POST, keep it as POST.

curl

```
curl -X POST https://api.linkedin.com/v2/adCreativesV2?ids=List(47770196) \
  -H 'Authorization: Bearer redacted' \
  -H 'X-RestLi-Method: BATCH_PARTIAL_UPDATE' \
  -H 'X-RestLi-Protocol-Version: 2.0.0' \
  --data ${"entities": {{"47770196": {"patch": {"$set": {"status":
"ACTIVE"}}}}}}'
```

2. Add the X-HTTP-Method-Override header using the original HTTP method `-H 'X-HTTP-Method-Override: POST'` for POSTs or `-H 'X-HTTP-Method-Override: PUT'` for PUTs.

curl

```
curl -X POST https://api.linkedin.com/v2/adCreativesV2?ids=List(47770196) \
  -H 'X-HTTP-Method-Override: POST' \
  -H 'Authorization: Bearer redacted' \
  -H 'X-RestLi-Method: BATCH_PARTIAL_UPDATE' \
  -H 'X-RestLi-Protocol-Version: 2.0.0' \
  --data ${"entities": {{"47770196": {"patch": {"$set": {"status":
"ACTIVE"}}}}}}'
```

3. Add the Content-Type header `-H 'Content-Type: multipart/mixed; boundary=xyz'`.

curl

```
curl -X POST https://api.linkedin.com/v2/adCreativesV2?ids=List(47770196) \
  -H 'X-HTTP-Method-Override: POST' \
  -H 'Content-Type: multipart/mixed; boundary=xyz' \
  -H 'Authorization: Bearer redacted' \
  -H 'X-RestLi-Method: BATCH_PARTIAL_UPDATE' \
  -H 'X-RestLi-Protocol-Version: 2.0.0' \
  --data ${"entities": {{"47770196": {"patch": {"$set": {"status":
"ACTIVE"}}}}}}'
```

4. Move the query string and original request body to body as two sections.

curl

```
curl -X POST https://api.linkedin.com/v2/adCreativesV2 \
  -H 'X-HTTP-Method-Override: POST' \
  -H 'Content-Type: multipart/mixed; boundary=xyz' \
  -H 'Authorization: Bearer redacted' \
  -H 'X-Restli-Protocol-Version: 2.0.0' \
```

```
  -H 'X-RestLi-Method:  BATCH_PARTIAL_UPDATE' \
  --data $'--xyz\r\nContent-Type: application/x-www-form-
urlencoded\r\n\r\nids=List(47770196)\r\n--xyz\r\n
        Content-Type: application/json\r\n\r\n{"entities": {"47770196":
{"patch": {"$set": {"status": "ACTIVE"}}}}}\r\n--xyz--'
```

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Rate Limiting

08/20/2025

To prevent abuse and ensure service stability, all API requests are rate limited. Rate limits specify the maximum number of API calls that can be made in a 24 hour period. These limits reset at midnight UTC every day.

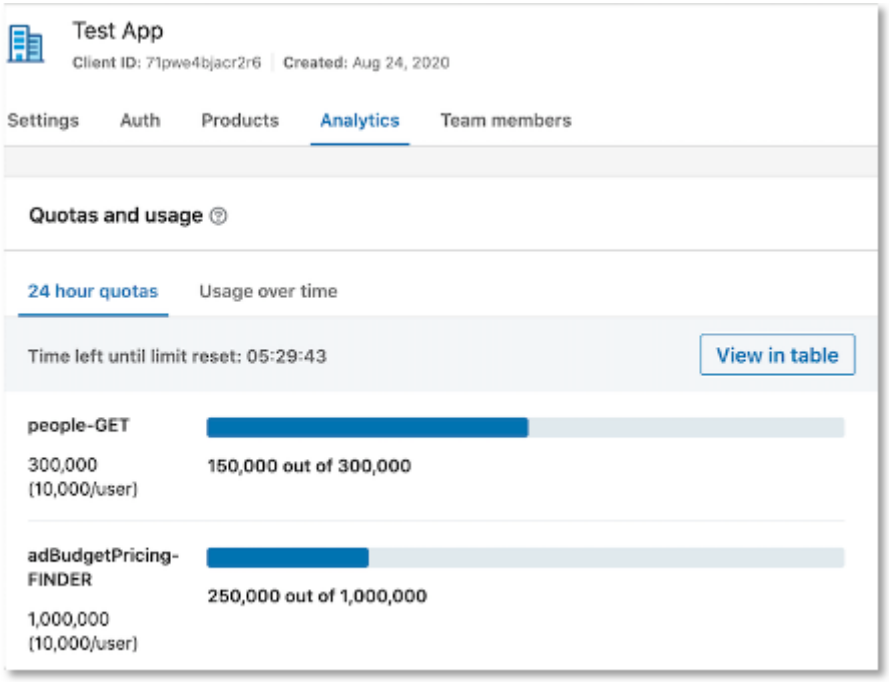
There are two kinds of limits that affect your application:

- **Application** — The total number of calls that your application can make in a day.
- **Member** — The total number of calls that a single member per application can make in a day.

> **Note**
>
> The term `Member` refers to a LinkedIn user whose token is used to initiate API calls from the developer application. For example, a partner is responsible for managing multiple members. This member-level designation indicates the permissible number of API calls the partner can initiate from their application on behalf of a member token.

Rate limited requests will receive a 429 response. In rare cases, LinkedIn may also return a 429 response as part of infrastructure protection. API service will return to normal automatically.

Your application's daily rate limit varies based on which API endpoint you are using. Standard rate limits are not published in documentation. You can look up the rate limit of any endpoint your app has access to through the Developer Portal. Select your app from the list and navigate to its Analytics tab. This page will only show usage and rate limits for endpoints you have made at least 1 request to today(UTC). If you want to look up a rate limit for an endpoint not listed in your app's Analytics page, make 1 test call to that endpoint and refresh the Analytics page.



Rate Limit Alert Notifications

Developer Admins will receive **email alerts** when their application exceeds **75%** of the assigned rate limit quota.

> ⓘ **Important**
>
> Alerts are triggered **only on application-level threshold breaches**, not on member-level or combined member+app-level breaches.

- **Alert Timing**: Alerts are **not sent in real-time**. There is an expected delay of approximately **1–2 hours** after the threshold is reached.
- **Additional Alerts**: If another API endpoint breaches the threshold later in the same day, a **new alert** will be sent, even if a previous alert was already delivered.

Request Methods

Article • 05/08/2023

LinkedIn's APIs have an expressive set of methods for interacting.

GET

The GET method requests a single, specific entity/object from a service. This method leverages the traditional HTTP GET method.

```
https

GET https://api.linkedin.com/v2/{service}/{resourceIdentifier}
```

GET_ALL

The GET_ALL method requests all entities/objects from a service. GET_ALL requests never provide resource identifiers to a service. This method uses the traditional HTTP GET method.

```
https

GET https://api.linkedin.com/v2/{service}
```

BATCH_GET

The BATCH_GET method requests more than one specific entity/object from a service. This method uses the traditional HTTP GET method.

```
https

GET https://api.linkedin.com/v2/{service}?
ids=List(resourceIdentifier_1,...,resourceIdentifier_n)
```

FINDER {finderName}

Conceptually, FINDER methods are similar to GET/BATCH_GET calls. The main difference between these methods is that you use FINDER queries when you don't have an

identifier to directly retrieve an entity. These methods use the HTTP GET method with a `q={finderName}` request parameter which identifies the type of query being made. FINDER methods can return zero, one, or more results, depending on the number of entities that match the query input.

```https
GET https://api.linkedin.com/v2/{service}?q={finderName}
```

CREATE

The CREATE method indicates to a service that it should use the information provided in the request body to create a new entity. This method uses the traditional HTTP POST method.

```https
POST https://api.linkedin.com/v2/{service}/{Request Body}
```

BATCH_CREATE

The BATCH_CREATE method indicates to a service that it should use the information provided in the request body to create multiple new entities. This method uses the traditional HTTP POST method.

```https
POST https://api.linkedin.com/v2/{service}/{Request Body}
```

UPDATE

The UPDATE method indicates to a service that it should use the information provided in the request body to overwrite the entire definition of an existing entity. This method uses the traditional HTTP PUT method.

```https
PUT https://api.linkedin.com/v2/{service}/{Request Body}
```

BATCH_UPDATE

The BATCH_UPDATE method indicates to a service that it should use the information provided in the request body to overwrite the entire definitions of multiple existing entities. This method uses the traditional HTTP `PUT` method.

https

```
PUT https://api.linkedin.com/v2/{service}/{Request Body}
```

PARTIAL_UPDATE

The PARTIAL_UPDATE method indicates to a service that it should use the information provided in the request body to update only specific portions of an existing entity rather than overwriting the entire definition of the entity. This method uses the traditional HTTP POST method.

> ⓘ **Note**
>
> For the server to differentiate between a PARTIAL_UPDATE and an UPDATE, you must include `X-Restli-Method: PARTIAL_UPDATE` as the header value in your request.

https

```
POST https://api.linkedin.com/v2/{service}/{Request Body}
```

BATCH_PARTIAL_UPDATE

The BATCH_PARTIAL_UPDATE method indicates to a service that it should use the multiple pieces of information provided in the request body to update specific portions of multiple specified entities, rather than overwriting them entirely. This method uses the traditional HTTP POST method.

> ⓘ **Note**
>
> For the server to differentiate between a `BATCH_PARTIAL_UPDATE` and a `BATCH_UPDATE`, you must include `X-Restli-Method: BATCH_PARTIAL_UPDATE` as the header value in your request.

https

```
POST https://api.linkedin.com/v2/{service}/{Request Body}
```

DELETE

The DELETE method indicates to a service that it should delete an identified object/entity. This method uses the traditional HTTP DELETE method.

https

```
DELETE https://api.linkedin.com/v2/{service}/{resourceIdentifier}
```

BATCH_DELETE

The BATCH_DELETE method indicates to a service that it should delete multiple identified objects/entities. This method uses the traditional HTTP DELETE method.

https

```
DELETE https://api.linkedin.com/v2/{service}?
ids=List({resourceIdentifier_1},...,{resourceIdentifier_n})
```

BATCH_FINDER {finderName}

Conceptually, BATCH_FINDER methods are similar to BATCH_GET calls. The main difference between these methods is that you use BATCH_FINDER queries when you don't have identifiers to directly retrieve entities. These methods use the HTTP GET method with a bq={batchFinderName} request parameter which identifies the type of query being made. BATCH_FINDER methods accept a list of filters set. Instead of calling multiple finders with different filter values, we call one BATCH_FINDER method with a list of filters. BATCH_FINDER methods can return zero, one, or more results, depending on the number of entities that match the query input.

https

```
GET https://api.linkedin.com/v2/{service}?bq={batchFinderName}
```

ACTION {actionName}

The ACTION method is a flexible method that does not specify any type of standard behavior. It uses the HTTP POST method, with a special `action={actionName}` request parameter, which identifies the specific type of action to take.

https

```
POST https://api.linkedin.com/v2/{service}?action={actionName}
```

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Response Decoration

Article • 02/05/2025

> ⚠ **Warning**
>
> **Deprecation Notice**
> The use of Response Decoration is deprecated in some versions of LMS APIs. Please refer the **Recent Changes** page for information on the specific API versions that are affected by this deprecation. You can also refer to **this video** to understand how to handle the deprecation of API decoration.

When you call an API, the response may contain URNs referencing the different types of objects provided by LinkedIn's services. These URNs are valuable on their own; however, there may be instances when you want to expand a URN and access the values associated with the entity.

Decoration is a mechanism in LinkedIn's APIs to fetch data belonging to a URN object without having to make an extra call to that object's API. Decoration uses a syntax very similar to LinkedIn's Field Projections.

See the following example of a service that returns URN references to another type of entity:

Sample Request

https

```
GET https://api.linkedin.com/v2/{service}/1234
```

Sample Response

JSON

```
{
    "id": 1234,
    "relatedEntity": "urn:li:relatedEntity:6789"
}
```

Rather than taking the `relatedEntity` URN value and making a second GET call to its parent service, you can use decoration to define how you'd like the `relatedEntity`

object to be expanded within the original API request. To do this, append the `~` character to the entity you wish to expand, and then provide the field projection in parentheses afterwards. For example:

Sample Request

https

```
GET https://api.linkedin.com/v2/{service}/1234&projection=(id,relatedEntity~($URN,foo,bar))
```

Sample Response

JSON

```
{
    "id": 1234,
    "relatedEntity": {
        "$URN": "urn:li:relatedEntity:6789",
        "bar": "bloop",
        "foo": "bleep"
    }
}
```

Decorating within Arrays/Lists

In some circumstances, a service might return a collection of URNs that are not all of the same type. In this case, provide multiple decoration instructions to tell the service how to deal with any potential URN type that is returned. See the following sample request and the list of different URN types that it returns within entities:

Sample Request

https

```
GET https://api.linkedin.com/v2/{service}/1234
```

Sample Data

JSON

```
{
    "entities": [
        "urn:li:foo:123",
        "urn:li:bar:234",
        "urn:li:baz:345"
    ],
    "id": 1234
}
```

To decorate each of the `foo`, `bar`, and `baz` URN types in this response, use the following projection syntax in your request:

Sample Request

https

```
GET https://api.linkedin.com/v2/{service}/1234?
projection=entities*~foo(a,b)~bar(c,d)~baz(e,f))
```

Sample Response

JSON

```
{
    "entities": [
        {
            "a": 1,
            "b": 2
        },
        {
            "c": 10,
            "d": 20
        },
        {
            "e": 100,
            "f": 200
        }
    ],
    "id": 1234
}
```

Sample Projections

Consider the following data model of a sample resource and review the below samples showing how to access various parts of this data in order to decorate.

JSON


```
{
    "person": {
        "current_position": {
            "company": "urn:li:company:1",
            "from": "2009",
            "job_title": "SWE"
        },
        "firstname": "First Name",
        "following_companies": [
            "urn:li:company:1",
            "urn:li:company:2"
        ],
        "lastname": "Last Name",
        "messages": {
            "urn:li:message:1": {
                "bcc": [
                    "urn:li:person:1",
                    "urn:li:person:39"
                ],
                "content": "xyz",
                "from": "urn:li:person:99"
            },
            "urn:li:message:2": {
                "bcc": [
                    "urn:li:person:1",
                    "urn:li:person:80"
                ],
                "content": "abc",
                "count": 1929
            }
        },
        "phone": "200-100-200",
        "position_history": [
            {
                "company": "urn:li:company:4888383",
                "from": "2006",
                "job_title": "SSE",
                "to": "2009"
            },
            {
                "company": "urn:li:company:39939",
                "from": "1999",
                "job_title": "SE",
                "to": "2006"
            }
        ]
    }
}
```


Accessing URN within a child object

`(person(current_position(company)))`

JSON

```
{
    "person": {
        "current_position": {
            "company": "urn:li:company:1"
        }
    }
}
```

Accessing URNs within an array of child objects

`(person(position_history(*(company))))`

JSON

```
{
    "person": {
        "position_history": [
            {
                "company": "urn:li:company:4888383"
            },
            {
                "company": "urn:li:company:39939"
            }
        ]
    }
}
```

Accessing URNs within an array of URNs

`(person(following_companies(*)))`

JSON

```
{
    "person": {
        "following_companies": [
            "urn:li:company:1",
            "urn:li:company:2"
        ]
    }
}
```

Selecting a single child field with a URN and expanding the URN

`(person(current_position(company~)))`

JSON


```
{
    "person": {
        "current_position": {
            "company": "urn:li:company:1",
            "company~": {
                "active": true,
                "administrators": [
                    "urn:li:person:6611647"
                ],
                "allEmployeesAsAdmins": false,
                "attributes": {
                    "disableThirdPartyNews": false,
                    "enableStatusUpdate": false,
                    "owns300x250RightAdSlot": false,
                    "paidCareers": false,
                    "paidGoldPlusCareers": false,
                    "paidPlatinumCareers": false,
                    "paidProducts": false,
                    "paidSilverCareers": false,
                    "paidSilverPlusCareers": false,
                    "showAnalytics": false,
                    "showBrandTree": false,
                    "showNews": true,
                    "showPastEmployees": false,
                    "silverProducts": false,
                    "useParentCareers": false
                },
                "companyId": 1,
                "companyType": "PUBLIC_COMPANY",
                "creationTime": 1375816467585,
                "creator": "urn:li:person:6611647",
                "dataVersion": 4,
                "descriptions": [
                    {
                        "description": "test test test testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest test",
                        "locale": "en_US"
                    }
                ],
                "employeeCountRange": "MYSELF_ONLY",
                "heroImage": {
```

```
                    "cropHeight": 220,
                    "cropWidth": 646,
                    "cropXPosition": 0,
                    "cropYPosition": 0,
                    "croppedImage":
"urn:li:media:/p/2/005/045/187/3d49ef0.png",
                    "height": 220,
                    "uncroppedImage":
"urn:li:media:/p/2/005/045/187/3b333d8.png",
                    "width": 646
                },
                "images": [
                    {
                        "originalMedia":
"urn:li:media:/p/2/005/045/188/1d5f329.png",
                        "type": "SQUARE_LOGO_LEGACY",
                        "urn": "urn:li:media:/p/2/005/045/188/1d5f329.png"
                    },
                    {
                        "originalMedia":
"urn:li:media:/p/2/005/045/188/24809b3.png",
                        "type": "LOGO_LEGACY",
                        "urn": "urn:li:media:/p/2/005/045/188/24809b3.png"
                    }
                ],
                "industries": [
                    "ACCOUNTING"
                ],
                "lastEditor": "urn:li:person:6611647",
                "lastModifiedTime": 1375816467585,
                "logo": "urn:li:media:/p/2/005/045/188/24809b3.png",
                "names": [
                    {
                        "active": true,
                        "locale": "en_US",
                        "name": "r-NQkR5TwJ",
                        "type": "CANONICAL"
                    }
                ],
                "organizationalEntity": "urn:li:organization:1",
                "squareLogo": "urn:li:media:/p/2/005/045/188/1d5f329.png",
                "status": "OPERATING",
                "stockSymbol": "",
                "twitterId": "",
                "universalName": "r-nqkr5twj",
                "websiteUrl": "https://www.whatismyreferer.com/"
            }
        }
    }
}
```

Selecting all child fields and fully expanding a child field containing a URN

`(person(current_position(*,company~)))`

JSON


```
{
    "person": {
        "current_position": {
            "company": "urn:li:company:1",
            "company~": {
                "active": true,
                "administrators": [
                    "urn:li:person:6611647"
                ],
                "allEmployeesAsAdmins": false,
                "attributes": {
                    "disableThirdPartyNews": false,
                    "enableStatusUpdate": false,
                    "owns300x250RightAdSlot": false,
                    "paidCareers": false,
                    "paidGoldPlusCareers": false,
                    "paidPlatinumCareers": false,
                    "paidProducts": false,
                    "paidSilverCareers": false,
                    "paidSilverPlusCareers": false,
                    "showAnalytics": false,
                    "showBrandTree": false,
                    "showNews": true,
                    "showPastEmployees": false,
                    "silverProducts": false,
                    "useParentCareers": false
                },
                "companyId": 1,
                "companyType": "PUBLIC_COMPANY",
                "creationTime": 1375816467585,
                "creator": "urn:li:person:6611647",
                "dataVersion": 4,
                "descriptions": [
                    {
                        "description": "test test test testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest testtest testtest testtest testtest testtest testtest testtest
testtest test",
                        "locale": "en_US"
                    }
                ],
                "employeeCountRange": "MYSELF_ONLY",
                "heroImage": {
```

```
                    "cropHeight": 220,
                    "cropWidth": 646,
                    "cropXPosition": 0,
                    "cropYPosition": 0,
                    "croppedImage":
"urn:li:media:/p/2/005/045/187/3d49ef0.png",
                    "height": 220,
                    "uncroppedImage":
"urn:li:media:/p/2/005/045/187/3b333d8.png",
                    "width": 646
                },
                "images": [
                    {
                        "originalMedia":
"urn:li:media:/p/2/005/045/188/1d5f329.png",
                        "type": "SQUARE_LOGO_LEGACY",
                        "urn": "urn:li:media:/p/2/005/045/188/1d5f329.png"
                    },
                    {
                        "originalMedia":
"urn:li:media:/p/2/005/045/188/24809b3.png",
                        "type": "LOGO_LEGACY",
                        "urn": "urn:li:media:/p/2/005/045/188/24809b3.png"
                    }
                ],
                "industries": [
                    "ACCOUNTING"
                ],
                "lastEditor": "urn:li:person:6611647",
                "lastModifiedTime": 1375816467585,
                "logo": "urn:li:media:/p/2/005/045/188/24809b3.png",
                "names": [
                    {
                        "active": true,
                        "locale": "en_US",
                        "name": "r-NQkR5TwJ",
                        "type": "CANONICAL"
                    }
                ],
                "organizationalEntity": "urn:li:organization:1",
                "squareLogo": "urn:li:media:/p/2/005/045/188/1d5f329.png",
                "status": "OPERATING",
                "stockSymbol": "",
                "twitterId": "",
                "universalName": "r-nqkr5twj",
                "websiteUrl": "https://www.whatismyreferer.com/"
            },
            "from": "2009",
            "job_title": "SWE"
        }
    }
}
```

Selecting all child fields and expanding child field with a URN for a single value

`(person(current_position(*,company~(vanityName))))`

JSON

```
{
    "person": {
        "current_position": {
            "company": "urn:li:company:1",
            "company~": {},
            "from": "2009",
            "job_title": "SWE"
        }
    }
}
```

Accessing complex type object

When a URN is a field's key and the value is an object, it is called a complex type object. In the following example, the `messages` field maps to an object with 2 fields that are `MessageUrns`. These 2 `MessageUrns` are keys that map to objects containing data such as `bcc`, `count`, and `content`. To access the `bcc` values for each `MessageUrn`, use the decoration `(person(messages()))(person(messages(*(from,bcc(*)))))`.

`(person(messages()))`

JSON

```
{
    "person": {
        "messages": {
            "urn:li:message:1": {
                "bcc": [
                    "urn:li:personr:1",
                    "urn:li:person:39"
                ],
                "content": "xyz",
                "from": "urn:li:person:99"
            },
            "urn:li:message:2": {
                "bcc": [
                    "urn:li:person:1",
                    "urn:li:person:80"
                ],
                "content": "abc",
```

```
                "count": 1929
            }
        }
    }
}
```

`(person(messages(*(from,bcc(*)))))`

JSON

```
{
    "person": {
        "messages": {
            "urn:li:message:1": {
                "bcc": [
                    "urn:li:person:1",
                    "urn:li:person:39"
                ],
                "from": "urn:li:person:99"
            },
            "urn:li:message:2": {
                "bcc": [
                    "urn:li:person:1",
                    "urn:li:person:80"
                ]
            }
        }
    }
}
```

Accessing all URNs within any array from a BATCH_GET call

Note that a BATCH_GET call returns a `results` field by default.

`(results(*(person(following_companies(*)))))`

Sample Data

JSON

```
{
    "results": {
        "123123": {
            "person": {
                "current_position": {
                    "company": "urn:li:company:1",
```

```
                    "from": "2009",
                    "job_title": "SWE"
                },
                "firstname": "First Name",
                "following_companies": [
                    "urn:li:company:1",
                    "urn:li:company:2"
                ],
                "lastname": "Last Name",
                "messages": {
                    "urn:li:message:1": {
                        "bcc": [
                            "urn:li:person:1",
                            "urn:li:person:39"
                        ],
                        "content": "xyz",
                        "from": "urn:li:person:99"
                    },
                    "urn:li:message:2": {
                        "bcc": [
                            "urn:li:person:1",
                            "urn:li:person:80"
                        ],
                        "content": "abc",
                        "count": 1929
                    }
                },
                "phone": "200-100-200",
                "position_history": [
                    {
                        "company": "urn:li:company:4888383",
                        "from": "2006",
                        "job_title": "SSE",
                        "to": "2009"
                    },
                    {
                        "company": "urn:li:company:39939",
                        "from": "1999",
                        "job_title": "SE",
                        "to": "2006"
                    }
                ]
            }
        }
    }
}
```


`(results(*(person(following_companies(*)))))`

JSON


```
{
    "results": {
```

```
        "123123": {
            "person": {
                "following_companies": [
                    "urn:li:company:1",
                    "urn:li:company:2"
                ]
            }
        }
    }
}
```

> **Note**
>
> Always use `entity~` in the projection parameter to make the expansion request. If the `entity` is expanded, the response returns `entity~`. If for some reason the `entity` cannot be expanded, the response returns `entity!`.

Rate Limiting

Response decoration makes calls to other services in order to resolve the requested, decorated entity. These calls are subject to rate limiting. It is possible for calls to return a 200 while the decoration call is rate limited.

The following example makes a request to the `/me` endpoint and uses response decoration to resolve the `digitalMediaAsset` URN in the `displayImage` field. The call to the `/me` endpoint is successful, but the decoration call to resolve `displayImage` is rate limited.

HTTP

```
GET https://api.linkedin.com/v2/me?projection=
(id,profilePicture(displayImage~(*)))
```

JSON

```
{
    "profilePicture": {
        "displayImage!": {
            "serviceErrorCode": 101,
            "message": "Resource level throttle limit for calls to this
resource is reached.",
            "status": 429
        },
        "displayImage": "urn:li:digitalmediaAsset:C4D03AQGFBHiaY1XXNA"
    },
```

```
    "id": "z6_nnTIGu-"
}
```


Feedback

Was this page helpful? Yes No

Provide product feedback

URNs and IDs

Article • 05/08/2023

URNs

URNs are used to represent foreign associations to an entity (persons, organizations, and so on) in an API. A URN is a string-based identifier with the format:

`urn:{namespace}:{entityType}:{id}`

For example:

- `urn:li:person:123`
- `urn:li:organization:456`
- `urn:li:sponsoredAccount:789`

URNs encode more information, including the entity type. Using the entity type, it's possible to dereference a URN and access the underlying data of the entity using a process called decoration.

URN values returned from LinkedIn's APIs are a maximum of 255 characters in length.

IDs

Simple integer IDs are returned to represent an object's primary key. Your application should support transforming URNs into IDs and vice versa.

IDs versus URNs

IDs are self-referencing identifiers similar to a primary key. URNs are Globally Unique Identifiers (GUID) used to represent an entity's foreign associations.

In the example below, we fetch details about a share and use a projection to limit the results to the `id` and `owner` fields. The `shares` API returns an `id` that is the primary key and an `owner` that is a foreign reference to the person who created the share in the form of a `URN`.

Resource Type	URN	ID
Share	`urn:li:share:1234`	1234

Resource Type	URN	ID
Person	`urn:li:person:-f_Ut43FoQ`	-f_Ut43FoQ

Sample Request

https

```
GET https://api.linkedin.com/v2/shares/1234?projection=(id,owner)
```

JSON

```
{
    "id": "1234",
    "owner": "urn:li:person:-f_Ut43FoQ"
}
```

URN Deconstruction

To get more information about the owner of the share, we can deconstruct the URN by removing the `urn:li:person` prefix and making an additional call to the `people` API. We need to deconstruct the URN and parse the ID because the `people` API expects an ID.

Sample Request

https

```
GET https://api.linkedin.com/v2/people/id=-f_Ut43FoQ?projection=
(id,localizedFirstName,localizedLastName)
```

JSON

```
{
    "id": "-f_Ut43FoQ",
    "localizedFirstName": "Dwight",
    "localizedLastName": "Schrute"
}
```

URN Decoration

To make this query more efficient, we can use decoration to expand the share owner URN and return the owner's first and last name in a single call. This saves an API call by not having to make a separate call to the `people` API.

Sample Request

https

```
GET https://api.linkedin.com/v2/shares/1234?projection=(id,owner~
(localizedFirstName,localizedLastName))
```

JSON

```
{
    "id": "1234",
    "owner": "urn:li:person:-f_Ut43FoQ",
    "owner~": {
        "localizedFirstName": "Schrute",
        "localizedLastName": "Dwight"
    }
}
```

URNs and Namespaces

For common URNs and their namespace conversions, see URN Namespaces.

Feedback

Webhooks

08/27/2025

Webhooks allow you to receive real-time HTTP notifications for subscribed events. This functionality is only available for applications with an approved use case for webhooks.

Webhook Registration and Validation

Notifications will only be sent to webhooks that are registered and validated.

To begin, register your webhook in the "Webhooks" tab of your application in the developer portal.

> **Note**
>
> The Webhooks tab is only enabled for applications with a use case for webhooks. For Lead Syncing use cases, webhook subscriptions must be created via the Lead Notification Subscriptions API and cannot be created via the UI.

Fields

Expand table

Field Name	Description	Required
applicationId	ID of the Developer Application being validated. Only sent for integrations with parent-child applications such as Apply Connect. Indicates which application's `clientSecret` to use for generating the challenge response.	False
challengeCode	Randomly generated type-4 UUID string created by LinkedIn.	True
challengeResponse	Hex-encoded HMACSHA256 signature for the `challengeCode` using its `clientSecret` as the secret key. `challengeResponse = Hex-encoded(HMACSHA256(challengeCode, clientSecret))`	True

Validating the Webhook Endpoint

LinkedIn validates the ownership of a webhook URL before it can be registered by an application. The validation flow leverages the application's `clientSecret` as the secret key

along with the universally-known HMACSHA256 algorithm to generate and validate the application's response to a challenge code.

1. Before initiating the validation flow, LinkedIn will generate a string that will act as the `challengeCode`.
2. LinkedIn will use the `challengeCode` as a query parameter to make an HTTP GET to your webhook endpoint. Integrations using parent-child applications will also receive an `applicationId` query parameter.

HTTP

```
GET https://webhooks.example.com/linkedin?challengeCode=890e4665-4dfe-4ab1-b689-ed553bceeed0
```

3. Your application must compute the challengeResponse (Hex-encoded HMACSHA256 signature for the challengeCode) using its clientSecret as the secret key. If you received an applicationId query parameter in the previous step, use the clientSecret associated with the challenged application. Return both the challengeCode and challengeResponse in a JSON payload with a 200 OK status within 3 seconds. See the example below. The response should have header "content-type" value set to "application/json".

```
challengeResponse = Hex-encoded(HMACSHA256(challengeCode, clientSecret))
```

JSON

```
{
 "challengeCode" : "890e4665-4dfe-4ab1-b689-ed553bceeed0",
 "challengeResponse" :
"27b1d19678542072a7f1d0ce845d0c78cec22567f413697e25648f44fa3d1514"
}
```

4. On receiving the validation response, LinkedIn will verify by computing the challenge response and comparing it with the `challengeResponse` returned by the app.
5. If the `challengeResponse` is successfully verified, then the webhook is ready to be used in subscriptions. If the verification fails, an error will be thrown with the message: `This URL did not pass the security challenge check`.

Re-validation and Blocked Endpoints

Webhook endpoints will be periodically re-validated after the initial validation every 2 hours. If the re-validation check fails 3 times in a row, the endpoint will move to a blocked state where events will no longer be sent.

Developers associated with that developer application will receive warning emails for any failed validation attempts. After re-validation checks fail 3 times in a row, an email will be sent notifying developers that the webhook has been blocked. The endpoint will also show as "Blocked" in the developer portal.

After resolving the issue, developers can manually kick off another validation check from the developer portal.

Requirements for Processing Notifications

Upon receiving notifications from LinkedIn, your webhook must fulfill the following requirements:

Verify the integrity of the push event and prevent malicious entities from impersonating LinkedIn as the sender

The POST request sent by LinkedIn will include a header called `X-LI-Signature`. The value of this header is the HMACSHA256 hash of the JSON-encoded string representation of the POST body prefixed by `hmacsha256=` and it is computed using your app's clientSecret. On receiving the push event, you must compute the signature on the POST body prefixed by `hmacsha256=` using the HMACSHA256 hashing algorithm and your clientSecret. Discard the event if your computed value does not match the value sent in the `X-LI-Signature` header.

Deduplicate notifications

A notification can occasionally be delivered multiple times to your webhook. Your webhook must be able to deduplicate notifications using the Notification ID included in the payload.

Respond with an HTTP response

For each notification delivered to your webhook, it must return a 2xx HTTP status code to indicate successful delivery of the notification to your webhook. Any other response code returned will be treated by LinkedIn as a failure.

Test with lambda/serverless functions on a cloud provider

We recommend testing webhooks processing using lambda/serverless functions on a cloud provider.

> **Note**
>
> - ngrok URIs are not supported.
> - All webhook URLs must use the HTTPS protocol. Non-HTTPS URLs are not supported.

LinkedIn API Clients

Article • 03/30/2023

Background

The Rest.li protocol defines 14 resource methods as a higher-level abstraction on top of the standard HTTP methods, each with its own standardized interface and intended semantics.

For example, to retrieve one or more entities, any one of these Rest.li methods can be implemented for a resource, depending on the use case: GET, BATCH_GET, GET_ALL, FINDER, or BATCH_FINDER. Each of those Rest.li methods would have its own standard request and response interfaces.

The Rest.li protocol also supports complex resource keys and query parameters, and it has a custom protocol for encoding them.

Nearly all of LinkedIn's APIs are built on the Rest.li framework with additional LinkedIn-specific details, which results in a robust yet complex protocol that can be challenging to implement correctly.

API Client Overview

To simplify development, we have released API client libraries in the following languages, each of which contains comprehensive documentation and examples in their corresponding GitHub repository:

Language	Installation and API Reference Docs	Examples
JavaScript	link	link
Python	link	link

Note that these client libraries do not support specific APIs - they provide an abstraction on top of the Rest.li protocol, making it easier to construct API requests to LinkedIn. This simplifies the process for you to build your higher-level clients for specific APIs.

Some of the key features of the LinkedIn API Client Libraries include:

- RestliClient with support for all Rest.li methods
- AuthClient with support for token creation/management
- Typed interfaces to enable code completion and inline documentation in IDEs

- Rest.li protocol 2.0 support
- Automatic key/query parameter encoding
- Support for versioned APIs
- Automatic query tunneling of requests, if required

Example

To illustrate the benefits of using these API clients, consider searching for ad accounts using our Advertising API. For a NodeJS application using an HTTP client library like axios, there are quite a few details to get right to perform a working request, as shown in the following sample code: base URL, resource prefix, auth and custom headers, and correct encoding of complex query parameters. The result can be difficult to write and maintain and doesn't account for edge cases requiring query tunneling.

JavaScript

```
const axios = require('axios');

const ACCESS_TOKEN = "AQX5…";
const API_VERSION = '202302';

// Define a generic function to search ad accounts by references
function searchForAdAccountsByReferences(
 references,
 count = 10,
 sortField = 'ID',
 sortOrder = 'ASCENDING'
) {
 return axios.request({
   baseURL: 'https://api.linkedin.com',
   url: '/rest/adAccounts',
   method: 'get',
   headers: {
     'Authorization': `Bearer ${ACCESS_TOKEN}`,
     'X-RestLi-Method': 'finder',
     'X-RestLi-Protocol-Version': '2.0.0',
     'Connection': 'Keep-Alive',
     'LinkedIn-Version': API_VERSION
   },
   params: {
     q: 'search',
     search: `(reference:
(values:List(${references.map(encodeURIComponent).join(',')})))`,
     count,
     sort: `(field:${sortField},order:${sortOrder})`
   },
   paramsSerializer: {
     // Can't use default encoding
     encode: (val) => val
```

```
    }
  }).then(response => response.data.elements);
}

// Search for ad accounts by a list of organizations, with up to 20 results
const orgs = ['urn:li:organization:123', 'urn:li:organization:456'];
searchForAdAccountsByReferences(orgs, 20).then(adAccounts => {
  console.log(adAccounts);
});
```

Compare that to this code utilizing the LinkedIn API client. The client abstracts much of the complexities of constructing the request. It allows passing in objects, arrays, and strings with special characters as is, making the code easy to write and maintain. Furthermore, it handles edge cases where query tunneling may be required.

JavaScript

```
const { RestliClient } = require('linkedin-api-client');

const restliClient = new RestliClient();
const ACCESS_TOKEN = "AQX5...";
const API_VERSION = '202302';

// Define a generic function to search ad accounts by references
function searchForAdAccountsByReferences(
  references,
  count = 10,
  sortField = 'ID',
  sortOrder = 'ASCENDING'
) {
  return restliClient.finder({
    resourcePath: '/adAccounts',
    finderName: 'search',
    queryParams: {
      search: {
        reference: {
          values: references
        },
      },
      count,
      sort: {
        field: sortField,
        order: sortOrder
      }
    },
    versionString: API_VERSION,
    accessToken: ACCESS_TOKEN
  }).then(response => response.data.elements);
}

// Search for ad accounts by a list of organizations, with up to 20 results
const orgs = ['urn:li:organization:123', 'urn:li:organization:456'];
```

```
searchForAdAccountsByReferences(orgs, 20).then(adAccounts => {
  console.log(adAccounts);
});
```

In summary, using these API clients can speed up and simplify the development process and make your application code maintainable and robust. If any issues arise, the corresponding Github repository is available for bug reports and enhancement suggestions.

Feedback

Was this page helpful? Yes No

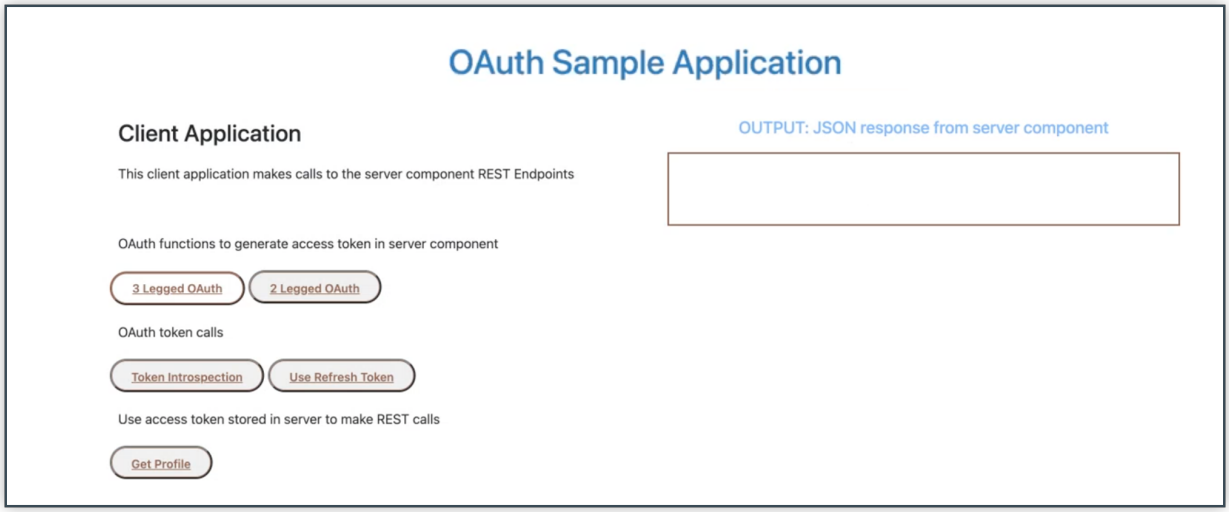
Provide product feedback | Get help at Microsoft Q&A

OAuth Sample Application

Article • 05/08/2023

Overview

The OAuth Sample Application is a ready-to-use code example that you can readily use to try out RESTful OAuth calls to the LinkedIn Authentication server. This version of the sample application is based on Java. The OAuth sample application shown below provides scalable and customizable code for your requirements as you begin API development with LinkedIn.



The sample application contains the client and server components you can use to manage your request calls to LinkedIn's APIs. The server creates and stores your access token and invokes APIs upon request from the client application.You can download or clone the OAuth sample application and try out these APIs as shown in the demo.

The OAuth sample application uses the following development tools:

- Spring Boot: provides the web server framework [https://spring.io/projects/spring-boot]
- LinkedIn OAuth 2.0: user authorization and API authentication
- Maven: app building and management
- Java: requires SE 7 or later versions for development

Prerequisites

- Register your application in the LinkedIn Developer Portal . After registration of the application, note down the Client ID and Client Secret for your future reference

- Add http://localhost:8080/login to the Authorized Redirect URLs under the **Authentication** section
- Configure your application build by installing and using Apache Maven
- Download the OAuth sample application from Sample Application Repository

Configure the OAuth sample application

Configure the client app:

1. Access the `application.properties` file. This file is located at `/client/src/main/resources/application.properties`
2. Enter the `server.port` and `SERVER_URL` in the noted fields:

```
JavaScript

server.port = <replace_with_required_port_no>
SERVER_URL = <replace_with_required_server_url>
```

3. Save your changes.

Configure the server app:

1. Access the `config.properties` file. This file is located at `/server/src/main/resources/config.properties`
2. Enter your client credential values in the noted fields:

```
JavaScript

clientId = <replace_with_client_id>
clientSecret = <replace_with_client_secret>
redirectUri = <replace_with_redirect_url_set_in_developer_portal>
scope = <replace_with_api_scope>
client_url = <replace_with_client_url>
```

3. Save your changes.

Start the OAuth sample application

To start the server:

1. Access the server folder.

2. Open the terminal/command prompt and run the following command to install dependencies: `mvn install`
3. Execute the following command to run the spring-boot server: `mvn spring-boot:run`

> ⓘ **Note**
>
> The server runs on `http://localhost:8080/`

To start the client:

1. Access the client folder.
2. Open the terminal/command prompt and run the following command to install dependencies: `mvn install`
3. Execute the following command to run the spring-boot server: `mvn spring-boot:run`

> **Note**: The client runs on `http://localhost:8989/`

Sample Application Demo

This demo shows an overall execution of the LinkedIn OAuth Sample Application. The demo contains clickable areas on the screen, which you can use to navigate.

> ⓘ **Note**
>
> You can reset the demo by refreshing the page.

Next Steps

If you are interested in exploring the sample application for Marketing, see Marketing Sample Application.

List of dependencies

Component Name	License	Linked	Modified
boot:spring-boot-starter-parent:2.5.2	Apache 2.0	Static	No
boot:spring-boot-starter-parent:2.5.2	Apache 2.0	Static	No
org.springframework.boot:spring-boot-starter-thymeleaf:2.2.2.RELEASE	Apache 2.0	Static	No
org.springframework.boot:spring-boot-devtools:2.6.0	Apache 2.0	Static	No
com.fasterxml.jackson.core:jackson-databind:2.13.0	Apache 2.0	Static	No
com.fasterxml.jackson.core:jackson-core:2.13.0	Apache 2.0	Static	No
org.springframework.boot:spring-boot-starter-web:2.5.2	Apache 2.0	Static	No
org.springframework.boot:spring-boot-starter-test:2.6.0	Apache 2.0	Static	No
org.springframework:spring-core:5.3.13	Apache 2.0	Static	No

Feedback

LinkedIn API Best Practices

Article • 05/08/2023

LinkedIn enables application development via a robust collection of APIs. See Getting Access for details on how to get access.

Use the following guidelines to build an application that members will trust to handle and secure their sensitive data.

- Application Development: Discover best practices for implementing authorization and error handling.
- Building Secure Applications: Use these guidelines to secure your application and members' data against threats such as phishing and cross-site request forgery.
- Breaking Changes: A breaking change is a change that may require you to make changes to your application to avoid disruption to your integration.
- Endpoint Catalog : This tool provides a view of the endpoints and permissions your application has access to. view, sort, and filter all the endpoints associated with an API product.
- LinkedIn Developer News : Stay on top of what's new with our portal, products, and programs.

Feedback

Best Practices for Application Development

Article • 05/08/2023

LinkedIn members are more comfortable trusting your application when you are transparent about how you will use their data. We recommend following these best practices to help your application deliver the most value.

Authentication

- Whenever possible, **remind the member that they are logged in** to your application by displaying their name, portrait, and/or account settings somewhere on the page.
- **Avoid having to log in multiple times**. When a member is integrated for multiple permissions, combine the permissions into a single request rather than asking the member to reauthenticate and grant consent each time.
- **Avoid generating an access token for each API call.Cache the member's access token** after they grant your application access, and do not re-authenticate the member unless they log out or the access token expires.
- Make sure you **allow the member to log out**, and when they do log out, ensure you destroy their access token and refresh token, as applicable.
- Validate the member access token using Token Introspection or by calling any API before making the access token call.
- Whenever the access token gets expired, make use of the refresh token, if applicable, to exchange for a new access token, unless the refresh token has also expired or been revoked.
- Reintroduce the member into the authentication flow only when both the access token and the refresh token have expired or been revoked.
- If you authorize the member through the JS SDK, do not send the member through the REST authorization flow. If you do, users will have to re-authorize your application. You can exchange the JS SDK token for an OAuth 2.0 REST access token if you want to make REST calls. Otherwise, use the JS SDK token to make calls with the JS SDK.

If a member authorizes your application through the REST workflow, it does not mean they are automatically logged in to the linkedin.com website. You should not assume that the member has access to resources that are on the LinkedIn website while using your application.

Error Handling and Logging

System Outages

Due to the nature of cloud APIs, LinkedIn's services are occasionally interrupted or temporarily unavailable for reasons outside of LinkedIn's control. Assume that any API call you make to LinkedIn or any third party could potentially fail. Always include error-handling logic in your requests. See the Errors page for API error codes and messages.

Errors

A `500 Internal Server Error` indicates that LinkedIn is experiencing an internal error. If you continue to receive server errors, record the following details:

- Request: `url`, `method`, `header`, e.g., `access_token`, `body`
- Response: `header`, e.g., `x-li-uuid`, `x-li-fabric`, `x-li-request-id`, `body`
- Your application configuration, e.g., `client_id`

If you continue to receive errors, reach out to your partner technical support channel, or view our Developer Support Knowledge.

Feedback

Best Practices for Secure Applications

09/03/2025

At LinkedIn, we take the privacy of our members very seriously. When we grant access to APIs, we expect developers to take member privacy just as seriously as we do. The LinkedIn platform uses permissions to protect and prevent abuse of our members' information. By using the OAuth 2.0 authentication protocol, we allow an application to access LinkedIn data while protecting members' credentials. Because of this protocol, members are ensured that applications on our platform are easy to use *and* protect their privacy and security.

> ⓘ **Note**
>
> You should always request the minimal scopes necessary and only request permissions that are needed for application functionality.

Ensure your application follows these best practices.

Access Tokens

Using access tokens, you can access a member's private information through the LinkedIn APIs. To keep access tokens safe:

- Do not store them in insecure or easily accessible locations. Client-side files, such as JavaScript or HTML files, should never be used to store sensitive information, as these can easily be accessed.
- Do not store access tokens in code files that can be decompiled, such as Native iOS, Android, or Windows Application code files.
- When making calls, always pass access tokens over a secure (HTTPS) connection.

API Key and Secret Key

Two pieces of identifiable information are required to make calls to the LinkedIn API: `Client ID` (Consumer Key/API key) and `Client Secret`.

The Client ID is a public identifier of your application. The Client Secret is confidential and should only be used to authenticate your application and make requests to LinkedIn's APIs.

Both the Client ID and Client Secret are needed to confirm your application's identity and it is critical that you do not expose your Client Secret. Follow these suggestions to keep the secret safe:

- Do not share your access tokens with anyone, and **do not** pass it in the URL when making API calls, or URI query-string parameters, or post in support forums, chat, etc.
- When creating a native mobile application, do not store it locally on a mobile device.
- Do not expose files such as JavaScript or HTML files in client-side code.
- Do not store it in files on a web server that can be viewed externally. For example, configuration files, include files, etc.
- Do not store it in log files or error messages.

Remember that when exchanging an OAuth 2.0 authorization code for an access token, `client_secret` is passed as part of the request. Make sure you **do not expose this request publicly**.
To reset or generate a new `client_secret` key, refer to the steps mentioned in the FAQ section.

Secure APIs

To prevent others from reading your requests and to prevent man-in-the-middle attacks, all OAuth 2.0 requests to our authentication servers must be done over HTTPS. Your application should also be hosted on a secure server, particularly for pages where a member enters private information (such as their password for your site) and for any URLs where you ask LinkedIn to redirect the member as part of the OAuth authorization flow.

Phishing Prevention

Cybercriminals often create websites that look and feel authentic but are really just replicas created to steal user credentials. Educate your users to look for signs to ensure they are entering credentials for a real LinkedIn application. Note that browsers may look different, and this may not always be enough to differentiate a legitimate site from a phishing site. Alert members not to enter credentials when in doubt and to contact you when they suspect suspicious activity.

LinkedIn has a DigiCert SHA2 Secure Server Certificate (padlock) that can be viewed prior to the URL in the browser. The base URL hostname is always "https://www.linkedin.com/ ...". Beware of URLs that try to mimic LinkedIn by using common misspellings or swapping similar characters such as "1" (one) for "l" (letter 'L'), e.g., "l1inkedIn.com/", or "linked1n.com/".

Cross-Site Request Forgery

To protect against CSRF attacks, during authorization, you must pass a `state` parameter. This should be a unique string value (for each request) that is unique, difficult to guess, and should not contain private or sensitive information.

Sample State Value

```https
state=760iz0bjh9gy71asfFqa
```

On successful authorization, the redirected URL should look like:

Sample Callback URL

```https
https://OAUTH2_REDIRECT_URI/?code=AUTH_CODE&state=760iz0bjh9gy71asfFqa
```

Make sure that the state parameter in the response matches the one you passed in your authorization request. If the state does not match, the request may be a result of CSRF and should be rejected.

Third-Party Libraries

When using a third-party library to interact with LinkedIn's APIs, make sure that the library is from a trusted source. Read reviews, look at the code, and do research to make sure the library is not malicious and does not behave unexpectedly.

LinkedIn **does not officially support third-party libraries**. Contact that library's development team if you have technical questions or concerns.

Frequently Asked Questions

1. What steps should developers take if the `client_secret` of their developer app has been compromised? or How can developers reset or rotate their `client_secret`?
   Answer: Execute the following steps to rotate the `client_secret`:
   a. In your Developer application, go to the **Auth** tab. Under the **Application Credentials** section, you'll find your **Client ID** and **Primary Client Secret**.
   b. Click on the **Generate a New Client Secret** button.
   c. In the confirmation dialog that appears, choose **Confirm**. Once confirmed, the current **Primary Client Secret** will become the **Secondary Client Secret** key. Your application will now have two active client secret keys.
   d. Replace the old client secret key in your configurations or code with the newly generated **Primary Client Secret** Key. Test the updated configurations/code to ensure everything works as expected.

e. After successful testing, navigate back to the **Application Credentials** section.
f. Click the Remove icon next to the **Secondary Client Secret** Key to delete it.

2. Can I immediately use the `client_id` and `client_secret` generated via the Provisioning API key to generate a token or load a widget?
   Answer: No. LinkedIn recommends waiting at least 5 minutes after creating a child app before using the associated `client_id` and `client_secret` for token generation or widget loading. This delay ensures that the newly provisioned app is fully registered and available across LinkedIn's systems, helping to prevent potential errors or incomplete initialization.

LinkedIn API Breaking Change Policy

Article • 05/08/2023

Overview

As our APIs evolve, LinkedIn will make reasonable efforts to notify you of breaking changes in advance with sufficient time to adapt to these changes.

> **ⓘ Important**
>
> We may make changes without prior notice if the change is considered non-breaking, or if it is a breaking change being made to address critical product bugs or legal, security, or privacy concerns.

Review this guide carefully to understand what kind of changes we consider to be breaking and non-breaking, and how to ensure that your application can adapt automatically to any change we categorize as non-breaking.

Definition

A **breaking change** is a change that may require you to make changes to your application in order to avoid disruption to your integration. The following are a few examples of changes we consider breaking:

- Changes to existing permission definitions
- Removal of an allowed parameter, request field or response field
- Addition of a required parameter or request field without default values
- Changes to the intended functionality of an endpoint. *For example, if a DELETE request previously used to archive the resource but now hard deletes the resource.*
- Introduction of a new validation

A **non-breaking change** is a change that you can adapt to at your own discretion and pace without disruption. In most cases, we will communicate non-breaking changes after they are already made. Ensure that your application is designed to be able to handle the following types of non-breaking changes without prior notice from LinkedIn:

- Addition of new endpoints
- Addition of new methods to existing endpoints
- Addition of new fields in the following scenarios:

    - New fields in responses
    - New optional request fields or parameters
    - New required request fields that have default values
- Addition of a new value returned for an existing text field
- Changes to the order of fields returned within a response
- Addition of an optional request header
- Removal of redundant request header
- Changes to the length of data returned within a field
- Changes to the overall response size
- Changes to error messages. *We do not recommend parsing error messages to perform business logic. Instead, you should only rely on HTTP response codes and error codes.*
- Fixes to HTTP response codes and error codes from incorrect code to correct code

Communication

Please look out for email communication from LinkedIn where we may notify you about breaking and non-breaking changes to our APIs. For breaking change notices, the email notice will usually include the future release date of the breaking change as well as instructions on what actions you need to take. To avoid disruption to your users, ensure that you perform the recommended actions by LinkedIn before the release date of the breaking change.

Feedback

LinkedIn Compliance Solutions Overview

LinkedIn provides the following Compliance API Guides for all monitoring, archiving, and management of communications for enterprises in regulated industries. The APIs will help your social interactions remain effective while ensuring compliance with corporate governance policies and major regulations.

Compliance Solutions for Sales Navigator

GET STARTED

Overview

Compliance Management API

Compliance Events API

Compliance - Sales Navigator FAQ

Compliance API Partner Program

GET STARTED

Overview

Compliance Snapshot API

Compliance Management API

Compliance Events API

Compliance API FAQ

General API Topics

TRAINING

Authentication

API Concepts

Breaking Change Policy

Best Practices

LinkedIn Consumer Solutions Platform

Article • 02/05/2025

The LinkedIn Consumer Solutions Platform enables sites and applications the power to enhance their sign-in experience using the world's largest professional network. The Consumer Solutions Platform contains APIs to Sign In with LinkedIn and Share on LinkedIn. Follow the links below to learn more about the Consumer Solutions Platform APIs.

- Sign In with LinkedIn using OpenID Connect
- Share on LinkedIn
- Add to Profile
- Plugins

Ready to start development? Refer to the links below for guidance on building your application.

- Authentication
- API Concepts
- Best Practices
- Breaking Change Policy
- Error Handling

Important

Notwithstanding anything to the contrary in the Microsoft Terms of Use and Microsoft Privacy Statement (please find the relevant links in the footer below), your use of the LinkedIn programmatic web APIs, software and other functionality and their associated tools and documentation that LinkedIn makes available to developers are governed by the **LinkedIn API Terms of Use** unless you have executed a separate signed partnership agreement, in which case that agreement shall apply.

Feedback

Was this page helpful? Yes No

Provide product feedback

LinkedIn Learning API and Technical Documentation

This homepage hosts LinkedIn Learning API and Technical Documentation. This documentation is for developers building integrations with LinkedIn Learning and technical administrators configuring LinkedIn Learning instances.

Overview and LinkedIn.com Blocking Policy

OVERVIEW

LinkedIn Learning Overview and LinkedIn.com Blocking Policy

APIs

HOW-TO GUIDE

Retrieve an Individual Learning Object

Retrieve Objects Based on Locale & Asset Type

Retrieve Objects Based on Criteria

Learning Activity Reports

xAPI Activity Webhooks

REFERENCE

learningAssets

learningClassifications

learningActivityReports

Authentication

HOW-TO GUIDE

SSO Overview

Configure your SSO

Configure your SSO with Okta

Configure your SSO with ADFS

Configure your SSO with Azure AD

Authenticate with Multiple SSO Methods

Switch from SSO to Non-SSO

Authenticate with Google

Authenticate with LTI

Authenticate Canvas with LinkedIn Learning via LTI

LMS Integrations (Non-Partner) - Packages & AICC Reporting

HOW-TO GUIDE

Schoox/UltiPro

Halogen/Saba TalentSpace

LMS365

Litmos

Other AICC Systems

LMS Integrations - Catalog Sync & AICC Reporting

HOW-TO GUIDE

Partner LMS Integrations Overview

Absorb

Adobe

CrossKnowledge

Digital Chalk

Docebo/Ceridian Dayforce

Infor

LMS Integrations - Catalog Sync & xAPI Reporting

HOW-TO GUIDE

360Learning

Bridge

Cornerstone

Degreed

Edcast

MindTickle

Saba

SumTotal

Totara

Workday

LMS Integrations - Defined Solutions

HOW-TO GUIDE

Blackboard

Knowledge Anywhere

Moodle

Pathgather

SuccessFactors

LinkedIn Learning Hub - Integration Guides

HOW-TO GUIDE

LinkedIn Learning Hub SFTP Guide

LMS and Content Integration Guide (It enabled)

LinkedIn Learning Career Hub + Workday HCM Guide

LinkedIn Learning for Library

HOW-TO GUIDE

LinkedIn Learning for Libraries Configuration

User and Attribute Management

HOW-TO GUIDE

Learner Management Overview

Add and Manage Learners in Bulk via CSV

Add Learner Data

Add Learners Automatically via SFTP

Add Learners Automatically via HTTPS

Revoke Learners in Bulk via CSV

Automate User Management with SCIM

Automate User Management with SCIM - OneLogin

Automate User Management with SCIM - Okta

Automate User Management with SCIM - Azure Active Directory

Reporting

HOW-TO GUIDE

Configure xAPI

Configure Reporting API

Reporting API FAQs

LinkedIn Marketing API Program

The LinkedIn Marketing Platform helps businesses grow by building technology to reach, engage, and convert professional audiences at scale. The platform offers APIs to create LinkedIn marketing campaigns, to report campaign performance, to manage leads, and to grow a company Page.

QUICKSTART
Quick Start

OVERVIEW
Getting API Access

HOW-TO...
Authentication

HOW-TO...
API Versioning

OVERVIEW
Data Storage Requirements

WHAT'S NEW
What's New

LinkedIn Marketing APIs

Advertising

With Advertising APIs, streamline your campaign management, audience targeting, and track...

- Campaign Management
- Reporting and Analytics

Event Management

Streamline event operations, amplify visibility, and drive meaningful connections within the professional landscape...

- Overview
- Events
- Event Management API Usecases

Community Management

Efficiently manage and engage with their LinkedIn communities, enhancing social media presence and...

- Overview
- Page Access Management
- Page and Content Analysis
- Shares & Social Stream
- Monitor Mentions of Your Brand
- Mention a Member or a Brand

Lead Sync

Seamlessly synchronize lead data captured through LinkedIn Lead Gen Forms with customer relationship...

- Overview
- Lead Sync
- Lead Sync API Use Cases

Matched Audiences

Reach specific audiences by uploading contact lists, website retargeting, or utilizing account targeting for more precise Ad...

- Overview
- DMP Segments
- DMP Segment Users
- DMP Segment Destinations
- DMP Segment Companies
- DMP Segment List
- Website Retargeting
- Ad Segments
- Matched Audiences API Use Cases

Audience Insights

Provides valuable data and analytics to advertisers, helping you better understand and target your audience on the...

- Overview
- Audience Insights API
- Audience Insights API Use Cases

Media Planning

Enables marketers to have a better understanding of investment delivery against valuable audiences and...

- Overview
- Media Planning API
- Manage Media Plans
- Media Planning API Use Cases

Conversions

Securely and accurately track and measure the impact of their LinkedIn ad campaigns by sending conversion event dat...

- Overview
- Conversions API
- Getting Access to Conversions API
- Deduplication
- Enabling Click IDs
- Google Tag Manager Server-Side Tagging
- Conversions API Payload Builder
- Use Cases - Conversions API

Company Intelligence

Gain insights into company level data for all companies reached via LinkedIn with engagement score, paid...

- Company Intelligence API

Support and Developer Resources

These resources can help you stay on top of new features, developer portal access, migrations, and deprecations.

Migrations

Stay on top of breaking changes related to migration.

Developer Portal

Stay on top of the developer offerings though our portal.

Developer Blog

Stay on top of the latest feature updates via available on our blog post.

Support

Need help? Reach out to Developer Support by submitting a Zendesk ticket.

Sales Navigator Application Platform Documentation

Article • 05/28/2025

LinkedIn Sales Navigator is a leading social selling tool that builds and nurtures customer relationships to lead to increased sales performance. By leveraging the power of LinkedIn's Sales Navigator, you can add exposure to sales leaders who are already engaged on LinkedIn and increase your product's engagement by integrating LinkedIn Sales Navigator seamlessly into your customers' workflow.

> **Note**
>
> We are not currently accepting new partners for access to the LinkedIn Sales Navigator API. We periodically review our onboarding capacity and will update this page if availability changes. Thank you for your understanding.

With the Sales Navigator Application Platform (SNAP) your customers can:

- target people and companies with a search experience that delivers relevant prospects
- understand key lead and company changes with suggestions customized for you
- engage with leads and prospects from directly within your application

Read on to learn more about the UI and API-based services that can help you maximize sales performance for your customers.

Display Services

Sales professionals have unique informational needs at different stages of the sales process. Display Services offers a robust set of UI modules for embedding within external applications via iframe or JavaScript SDK to enrich workflow experiences with LinkedIn identity and insights. Display Services modules allow viewing LinkedIn profiles and accounts, displaying recent activity and posts, and connecting and sending messages (including InMail) to leads. External applications can also enrich UI presentations with LinkedIn profile photos and links to Sales Navigator profiles with the Profile Associations API.

To learn more about Display Services, visit the Display Services Overview.

Analytics Services

Analytics Services offers a bulk export API to retrieve activities performed by sales teams, as well as other metadata like a seat holder's daily Social Selling Index, total connections, total leads saved, and other metrics.

To learn more about Analytics Services, visit the Analytics Services Overview.

Sync Services

Sales Navigator Sync Services offers APIs that enable richer integrations built on top of CRM Sync. By leveraging CRM Sync matches between CRM records and LinkedIn, CRM applications can enable new data integrity workflows and UI experiences for CRM users.

To learn more about Sync Services, visit the Sync Services Overview.

> **Important**
>
> Notwithstanding anything to the contrary in the Microsoft Terms of Use and Microsoft Privacy Statement (please find the relevant links in the footer below), your use of the LinkedIn programmatic web APIs, software and other functionality and their associated tools and documentation that LinkedIn makes available to developers are governed by the LinkedIn SNAP Terms of Use unless you have executed a separate signed partnership agreement, in which case that agreement shall apply.

LinkedIn Talent Solutions Overview

LinkedIn Talent Solutions enhances candidate sourcing and recruiting experiences for ATSs and applications by leveraging the world's largest professional network. Follow the links below to learn more about the LinkedIn Talent Solutions APIs.

HOW-TO...
Authentication

HOW-TO...
API Versioning

WHAT'S NEW
What's New

LinkedIn Talent APIs

Apply Connect

Apply Connect is an ATS integration from LinkedIn that improves the candidate experience by allowing job...

Apply Connect

Apply with LinkedIn

Apply with LinkedIn (AWLI) aims to bring a shorter and more seamless job application experience to candidates wh...

Apply with LinkedIn

Easy Apply

Easy Apply enables you, the job poster, to retrieve applicants that applied to your jobs on LinkedIn.

Easy Apply

Job Posting

Jobs that are already created externally on ATS, company site, or job board can be automatically ingested by...

Job Posting API

CRM Connect

CRM Connect synchronizes candidate information between your Candidate Relationship Management system and...

CRM Connect

Recruitment System Connect

Recruiter System Connect synchronizes customers' candidate information between your ATS and LinkedIn...

Recruitment System Connect

Support and Developer Resources

These resources can help you stay on top of new features, developer portal access, migrations, and deprecations.

Migrations

Stay on top of breaking changes related to migration.

Developer Portal

Stay on top of the developer offerings though our portal.

Developer Blog

Stay on top of the latest feature updates via available on our blog post.

Support

Guide for integration partners to leverage LinkedIn's support ecosystem.

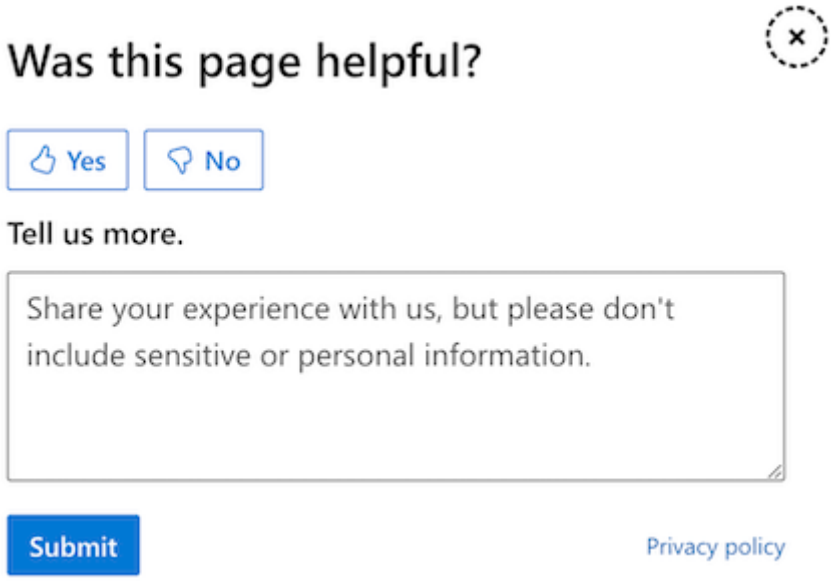
How to Provide Documentation Feedback to LinkedIn

09/24/2025

LinkedIn values your feedback and reviews it regularly to make timely improvements. By providing clear feedback, you help improve documentation quality and drive changes you want to see. Reach out to our documentation team directly using the Feedback button at the top right corner of every page.

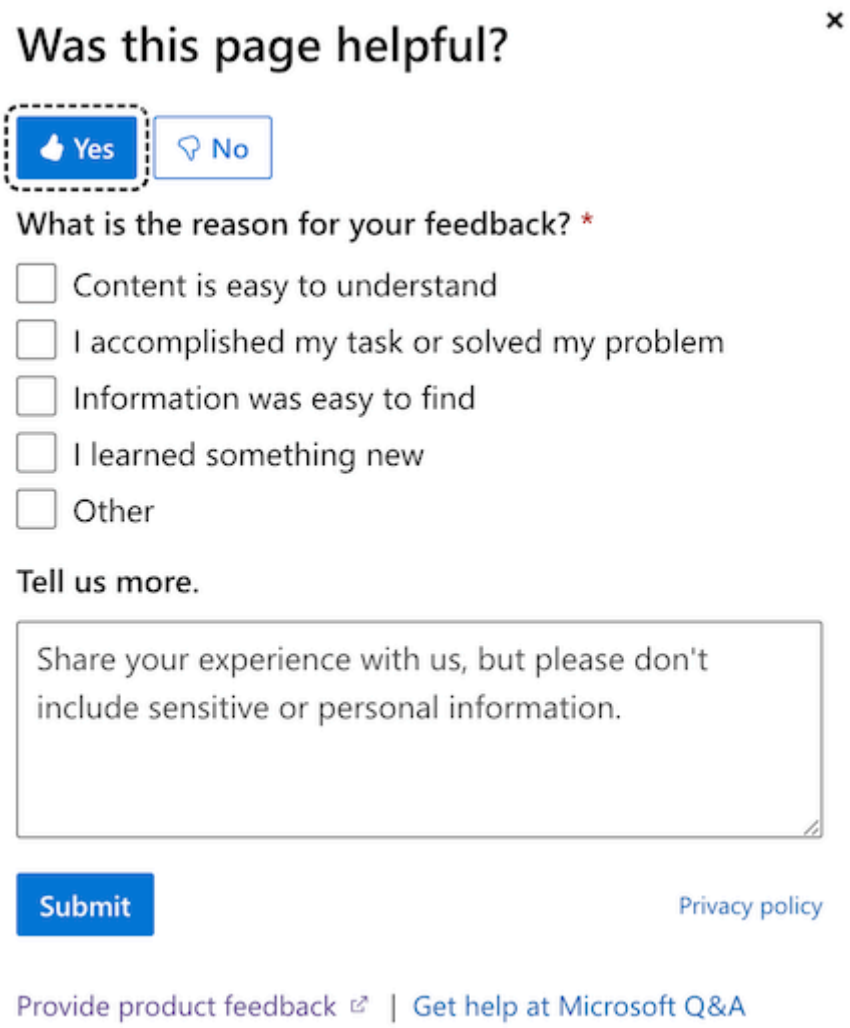
Steps to provide documentation feedback are given below:

1. Use the **Was this page helpful?** prompt that appears at the top right corner of the page.



2. Click the **thumbs-up** or **thumbs-down** button to indicate whether the content was helpful.

   The feedback options are displayed as shown below:



3. Select the reason for your feedback. This field is required.

4. Enter your comments in the **Tell us more** section to provide additional details.

> **Important**
>
> Please provide detailed feedback about the content changes you want to see. This helps us respond quickly and prioritize fixes. Do not include sensitive or personal information in your comments.

5. Click **Submit** to send the feedback to our documentation team.

Once we receive your feedback, our team will evaluate it and prioritize changes based on urgency and relevance.

> **Note**

To provide product-related feedback, click **Provide product feedback** in the feedback prompt. This will redirect you to the **LinkedIn Developer Support Portal**, where you can submit a ticket directly to our product teams.