LinkedIn Consumer Solutions Platform

Article • 06/30/2023

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

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

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)

Article • 11/30/2023

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.

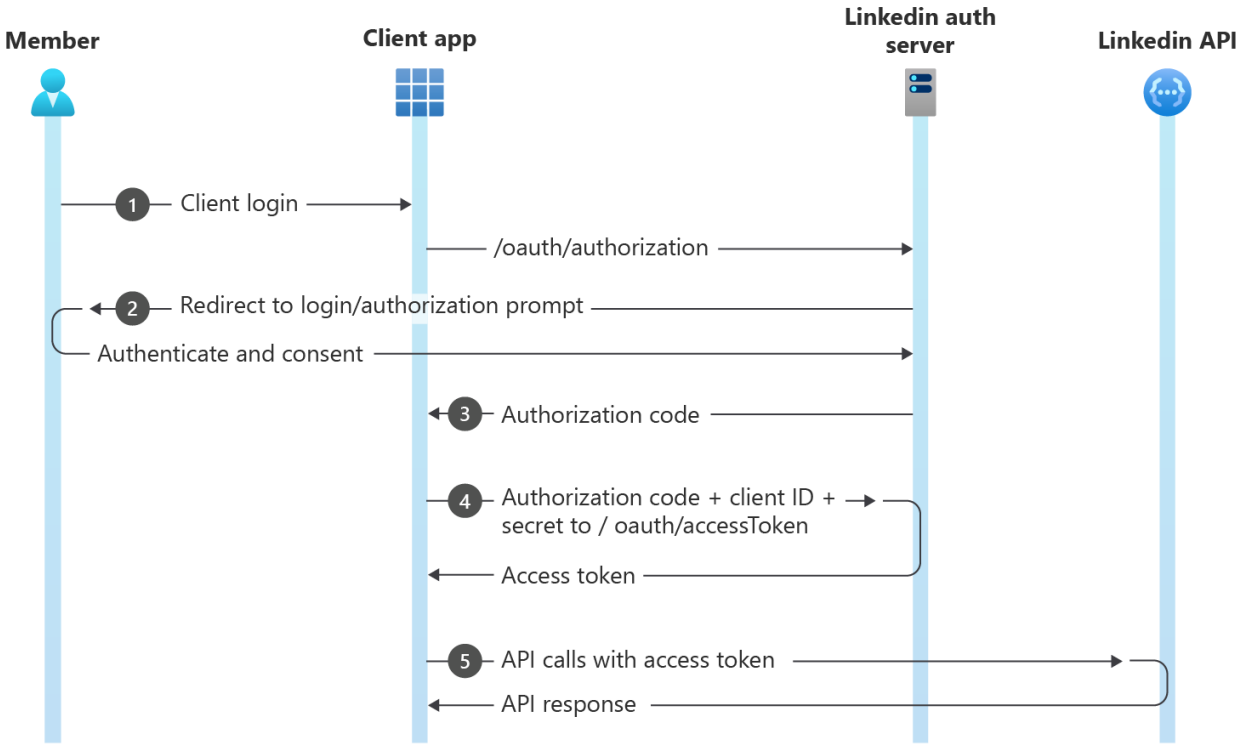
This approval process ensures that LinkedIn members are aware of what level of detail an application may access or action it may perform on their behalf.

If multiple scopes are requested, the user must be consent to all of them and may not select 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 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**.

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

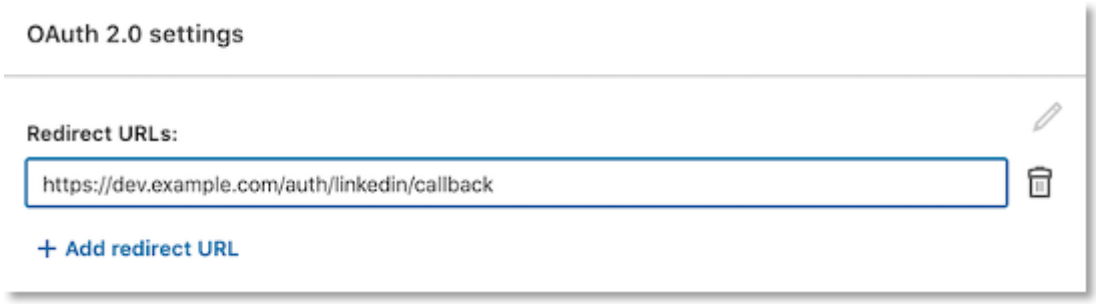
1. Select your application in the LinkedIn Developer Portal.
2. Click the **Auth** tab to view your application credentials.
3. Add the redirect (callback) URL via `HTTPS` to your server.

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

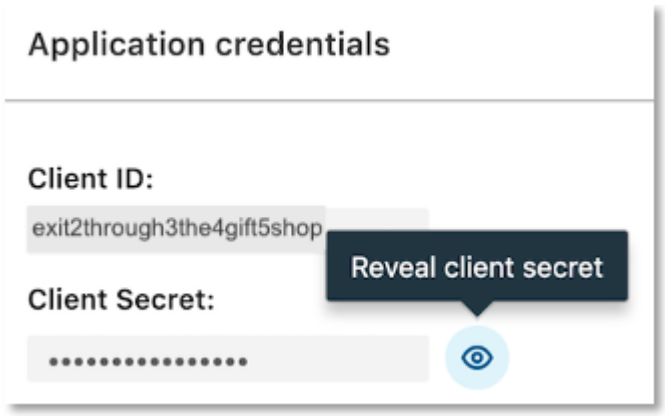
- `https://dev.example.com/auth/linkedin/callback`
- *not* `/auth/linkedin/callback`

- 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**

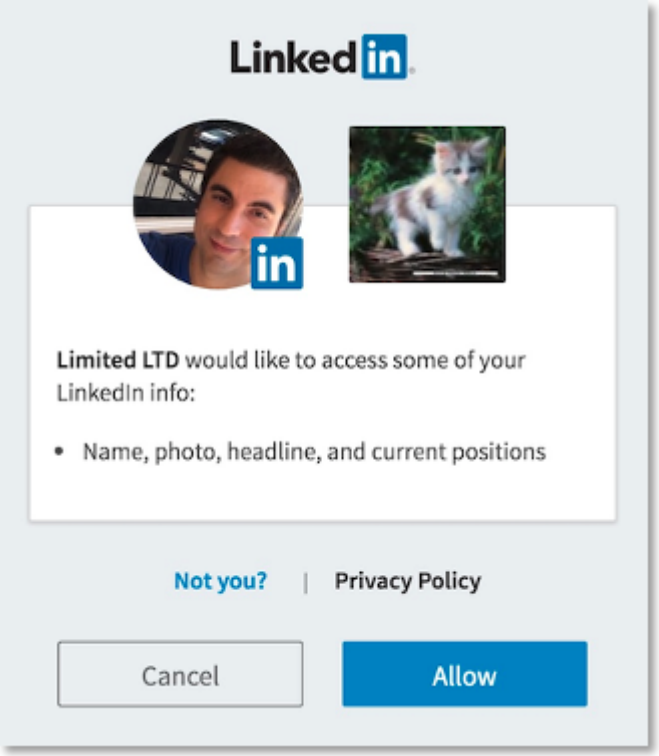
Do not share your *Client Secret* value with anyone, and **do not** pass it in the URL when making API calls, or URI query-string parameters, or post in support forums, chat, etc.

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.

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

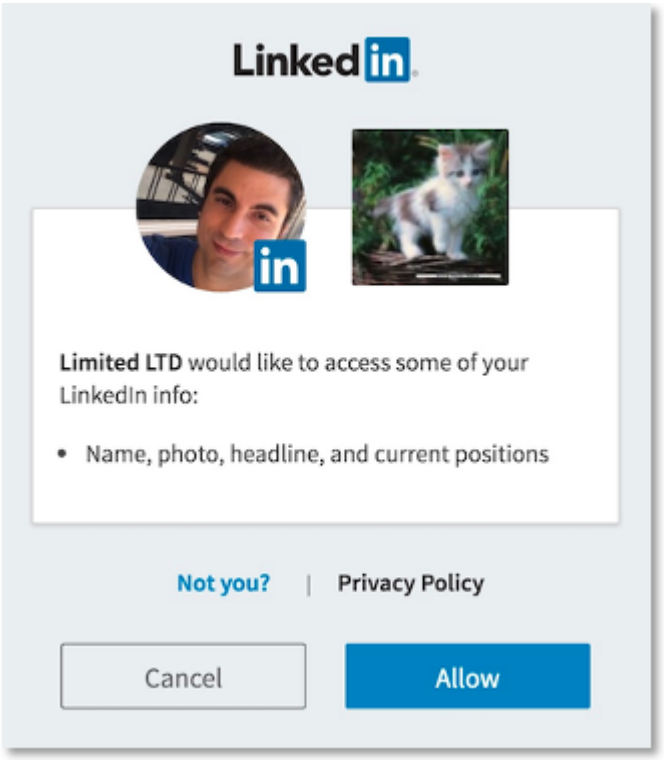
The scopes available to your app depend on which Products or Partner Programs your app has access to. This information is available in the Developer Portal. Your app's Auth tab will show current scopes available. You can apply for new Products under the Products tab. If approved, your app will have access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

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 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, refer 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.

Parameter	Type	Description
expires_in	int	The number of seconds remaining until the token expires. Currently, all access tokens are issued with a 60-day lifespan.
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 its resolution for 3-legged OAuth. If you wish to view the standard HTTP status codes and its 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.
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 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 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 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 mandatory parameter.	Add grant_type as "authorization_code" in the request.

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
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 "client_secret" is missing"	Client Secret in the request is missing. It is 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 Authorization code is sent as part of the request"	Authorization code expired and re-authenticate member to generate new authorization code and pass the fresh authorization code to exchange for access token.

Feedback

Client Credential Flow (2-legged OAuth)

Article • 05/08/2023

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.

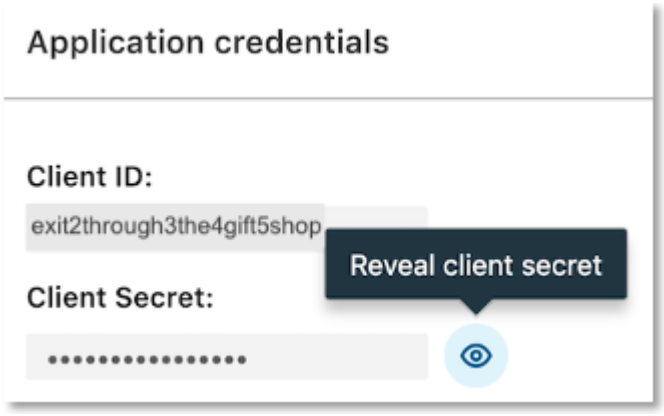
> **Important**
>
> 2-legged OAuth authentication is not available for **Marketing APIs**

> **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
```

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	Yes

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

Sample Request (Secure Approach)

```
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

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

Refresh Tokens with OAuth 2.0

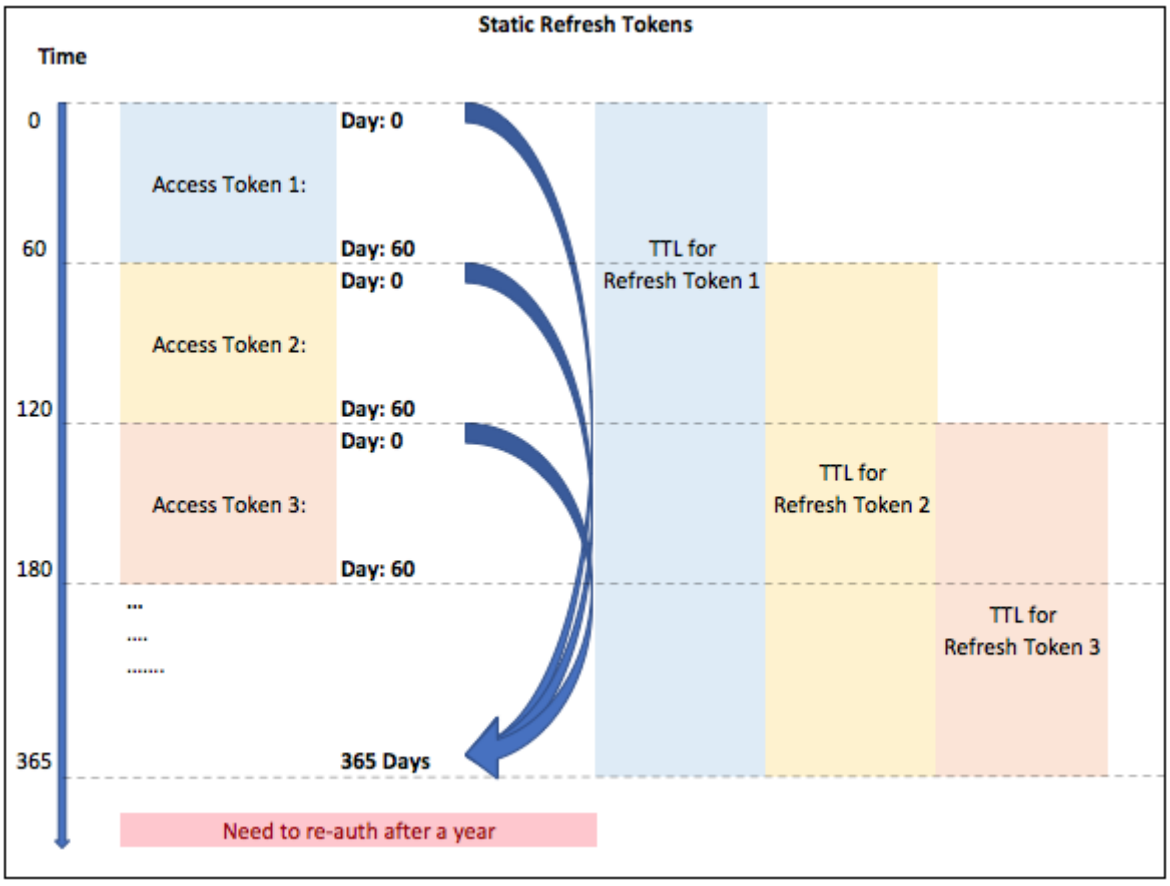
Article • 05/08/2023

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.

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-
wvaKrkMLfxxaB1O4poRg2svCWWgwhebQhqrETYlLikJJMgRAvH1ostjXd3DP3BtwzCGeTQ7K9vvA
qfQK5iG_eyS-q-
y8WNt2SnZKZumGaeUw_zKqtgCQavfEVCddKHcHLaLPGVUvjCH_KW0DJIdUMXd90kWqwuw3UKH27k
i5raFDPuMyQXLYxkqq4mYU-IUuZRwq1pcrYp1Vv-
ltbA_svUxGt_xeWeSxKkmgivY_DlT3jQylL44q36ybGBSbaFn-
```

```
UU7zzio4EmOzdmm2tlGwG7dDeivdPDsGbj5ig",
  "expires_in": 86400,
  "refresh_token": "AQWAft_WjYZKwuWXLC5hQlghgTam-tuT8CvFej9-
XxGyqeER_7jTr8HmjiGjqil13i7gMFjyDxh1g7C_G1gyTZmfcD0Bo2oEHofNAkr_76mSk84sppsG
bygwW-5oLsb_OH_EXADPIFo0kppznrK55VMIBv_d7SINunt-
7DtXCRAv0YnET5KroQOlmAhc1_HwW68EZniFw1YnB2dgDSxCkXnrfHYq7h63w0hjFXmgrdxeeAuO
HBHnFFYHOWWjI8sLLenPy_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
```

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_kQNOxsLRL
ZWnAMzHMm81S0yQlkBYicw&client_id=861hhm46p48to2&client_secret=gPecS7yqHkyySh
vR
```

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

JSON

```
{
  "access_token": "BBBB2kXITHELmWblJigbHEuoFdfRhOwGA0QNnumBI8XOVSs0HtOHEU-
wvaKrkMLfxxaB1O4poRg2svCWWgwhebQhqrETYlLikJJMgRAvH1ostjXd3DP3BtwzCGeTQ7K9vvA
qfQK5iG_eyS-q-
y8WNt2SnZKZumGaeUw_zKqtgCQavfEVCddKHcHLaLPGVUvjCH_KW0DJIdUMXd90kWqwuw3UKH27k
i5raFDPuMyQXLYxkqq4mYU-IUuZRwq1pcrYp1Vv-
ltbA_svUxGt_xeWeSxKkmgivY_DlT3jQylL44q36ybGBSbaFn-
UU7zzio4EmOzdmm2tlGwG7dDeivdPDsGbj5ig",
  "expires_in": 86400,
  "refresh_token": "AQWAft_WjYZKwuWXLC5hQlghgTam-tuT8CvFej9-
XxGyqeER_7jTr8HmjiGjqil13i7gMFjyDxh1g7C_G1gyTZmfcD0Bo2oEHofNAkr_76mSk84sppsG
bygwW-5oLsb_OH_EXADPIFo0kppznrK55VMIBv_d7SINunt-
7DtXCRAv0YnET5KroQOlmAhc1_HwW68EZniFw1YnB2dgDSxCkXnrfHYq7h63w0hjFXmgrdxeeAuO
HBHnFFYHOWWjI8sLenPy_EBrgYIitXsAkLUGvZXlCjAWl-
W459feNjHZ0SIsyTVwzAQtl5lmw1ht08z5Du-RiQahQE0sv89eimHVg9VSNOaTvw",
  "refresh_token_expires_in": 439200,
  "scope":"r_basicprofile"
}
```

API Error Details

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.

HTTP STATUS CODE	ERROR MESSAGE	ERROR DESCRIPTION	RESOLUTION
400	invalid_request "A required parameter "grant_type" is missing"	Grant type in the request is missing. It is mandatory parameter.	Add grant_type as "refresh_token" in the request.
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.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Developer Portal Tools

Article • 05/08/2023

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

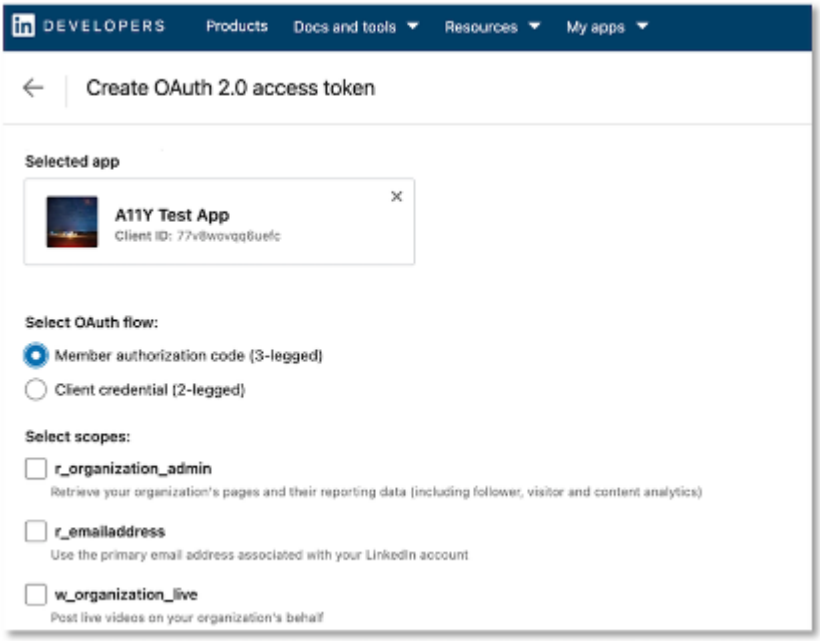
Generate a Token in the Developer Portal

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.

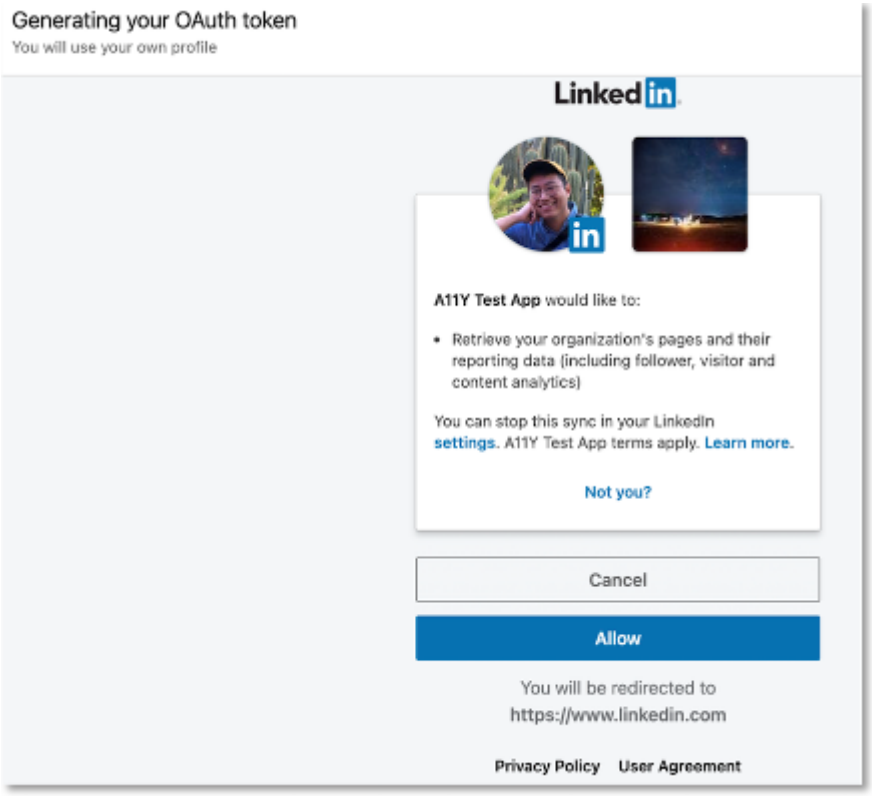
2. Select the app you'd like to generate a token for.

3. Select OAuth flow and permission scopes.



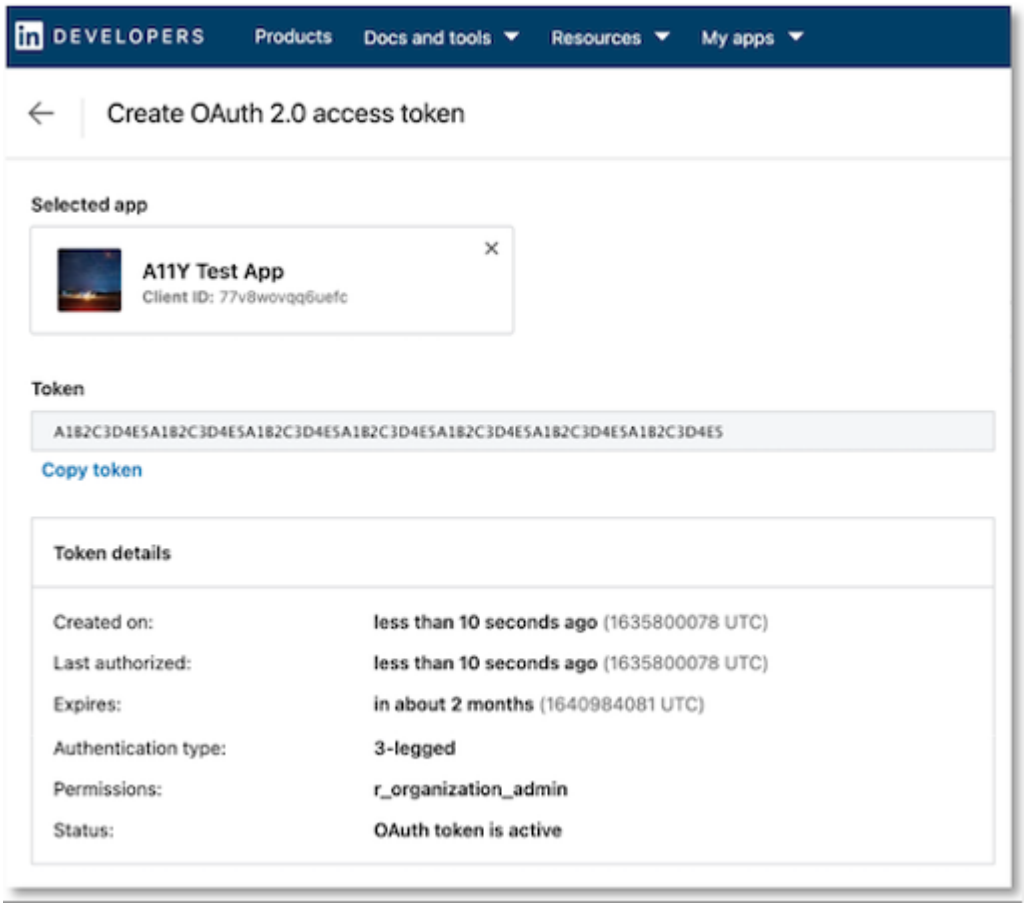
4. Member approval

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

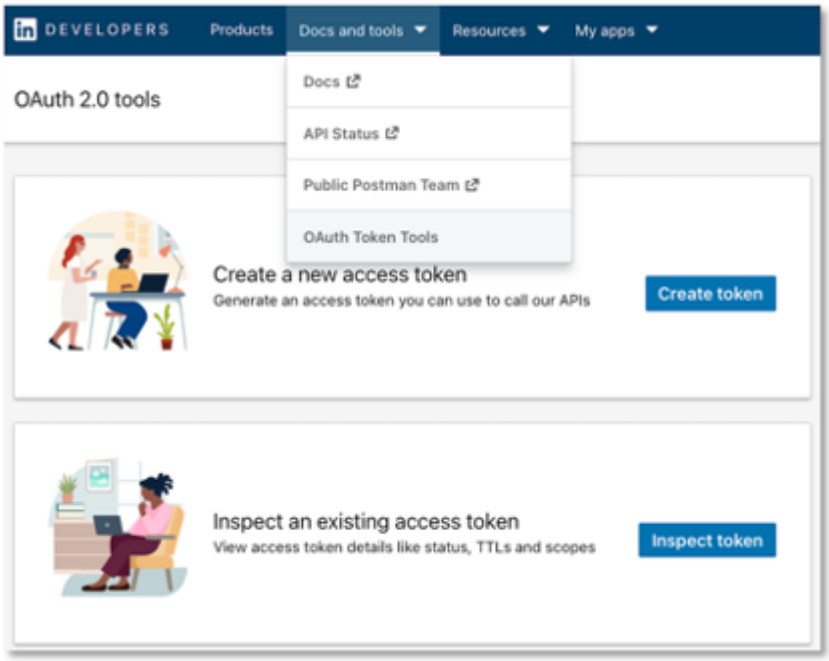


5. Token Generation

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.



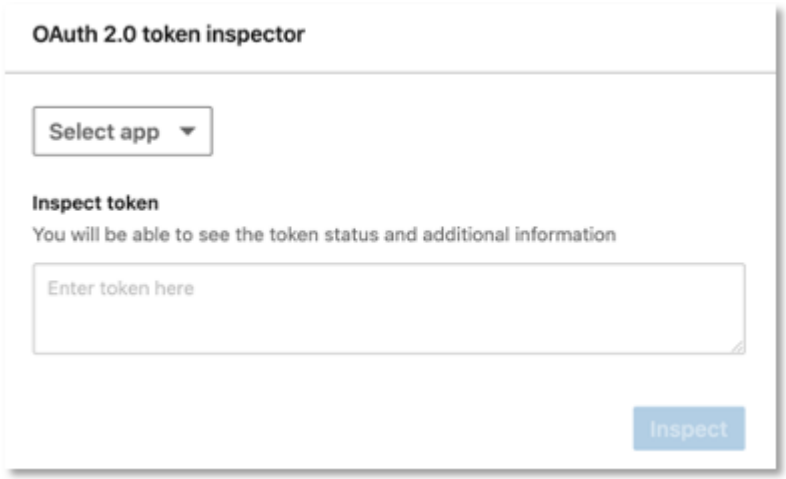
Should you wish to verify this an existing token, the Token Inspector tool is available on the same page as the token generator.



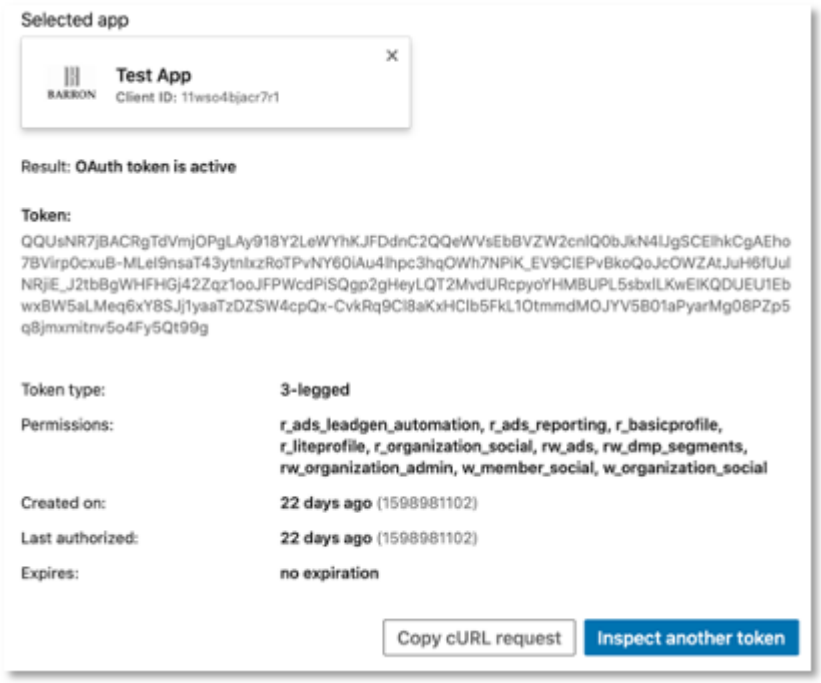
Developer Portal Token Inspector

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.



Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Token Introspection

Article • 12/01/2023

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

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

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
expires_at	long	Optional. Epoch time in seconds, indicating when this token will expire

Field	Type	Description
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.

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

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

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

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

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

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

Response Decoration

Article • 09/14/2023

> ⚠ **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.

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

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

Rate Limiting

Article • 09/06/2023

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.

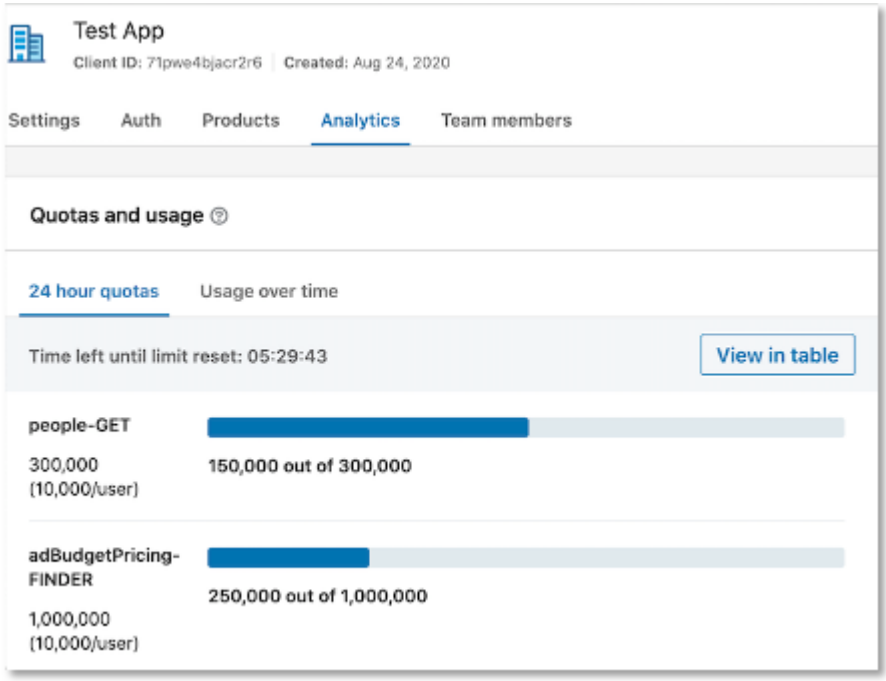
There are two kinds of limits that affect your application:

- **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.



Feedback

Was this page helpful? Yes No

Provide product feedback

Error Handling

Article • 05/08/2023

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:

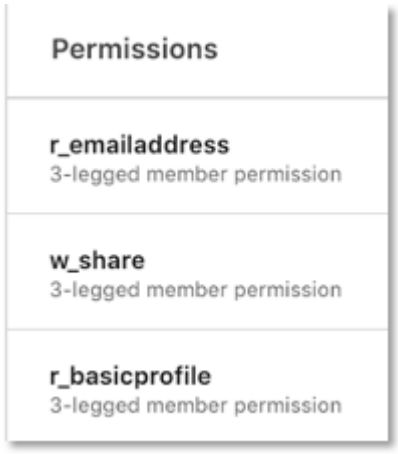
Error Type	Fix
Unknown authentication schema	Unrecognized authentication header schema. Make sure the authentication header follows the format `Authorization: Bearer (your access token)`
Empty OAuth2 access token	The authentication header is missing or empty. Make sure the authentication header follows 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.

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

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

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

Article • 05/08/2023

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!**

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.

Feedback

Error Handling

Article • 05/08/2023

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:

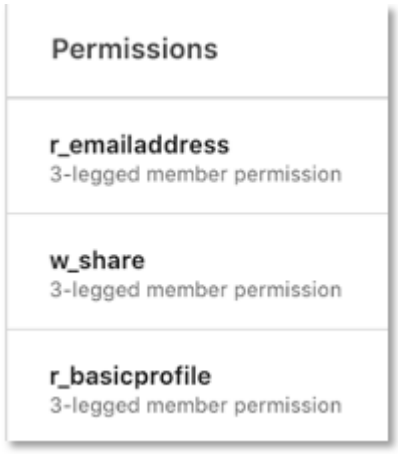
Error Type	Fix
Unknown authentication schema	Unrecognized authentication header schema. Make sure the authentication header follows the format `Authorization: Bearer (your access token)`
Empty OAuth2 access token	The authentication header is missing or empty. Make sure the authentication header follows 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.

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

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

Sign In with LinkedIn using OpenID Connect

Article • 08/10/2023

Overview

We are now offering a way for your apps to authenticate members using OpenID Connect.

What is OpenID Connect

OpenID Connect (OIDC) is an identity layer built on top of OAuth 2.0, enabling applications to authenticate members and obtain lite profile information about the member. OIDC will simplify the partner integration onboarding experience and eliminate dependencies on making additional API calls to find who the authenticated member is.

> **Note**
>
> Keep in mind Sign In with LinkedIn using OpenID Connect does not verify user identities and should not be marketed as such.

Getting Started

Authenticating Members

New members logging in to your service for the first time will need to follow the Authenticating with OAuth 2.0 Guide. When requesting the authorization code in Step 2 of the OAuth 2.0 Guide, make sure you use the OpenID scope `openid` to get the ID Token. We are also introducing new scopes `profile` and `email`.

Permission Name	Description
openid	Required to indicate the application wants to use OIDC to authenticate the member.
profile	Required to retrieve the member's lite profile including their id, name, and profile picture.

Permission Name	Description
email	Required to retrieve the member's email address.

After successful authentication, you will receive the member's access token and ID token.

If your application does not have these permissions provisioned, you can request access through the Developer Portal. Select your app from My Apps, navigate to the Products tab, and request the Sign in with LinkedIn using OpenID Connect product.

Retrieving Member Profiles Using ID Tokens

The primary extension that OIDC adds to OAuth 2.0 is enabling members to be Authenticated using the ID Token data structure. The ID Token is a security token that contains Claims about the Authentication of a member by an Authorization Server when using a Client, and potentially other requested Claims. The ID Token is represented as a JSON Web Token (JWT). With OIDC, you are now able to extract the member details from the ID Token.

ID Token Payload

Permission Name	Description
iss	The authorization server's Identifier of the response https://www.linkedin.com
sub	User identifier
aud	client_id of Relying party/caller Application guarantees user authentication for itself.
iat	ID Token issue time
exp	This is the expiration time of the token. The relying party processing this token should reject it once the expiration time is reached.

Validating ID Tokens

ID Tokens can be validated using the metadata provided by discovery document which contains OAuth endpoints, public keys and claims.

```
JSON
```

```
{
    "issuer": "https://www.linkedin.com",
    "authorization_endpoint":
"https://www.linkedin.com/oauth/v2/authorization",
    "token_endpoint": "https://www.linkedin.com/oauth/v2/accessToken",
    "userinfo_endpoint": "https://api.linkedin.com/v2/userinfo",
    "jwks_uri": "https://www.linkedin.com/oauth/openid/jwks",
    "response_types_supported": [
        "code"
    ],
    "subject_types_supported": [
        "pairwise"
    ],
    "id_token_signing_alg_values_supported": [
        "RS256"
    ],
    "scopes_supported": [
        "openid",
        "profile",
        "email"
    ],
    "claims_supported": [
        "iss",
        "aud",
        "iat",
        "exp",
        "sub",
        "name",
        "given_name",
        "family_name",
        "picture",
        "email",
        "email_verified",
        "locale"
    ]
}
```


JWKS_URI provided in the discovery document, can be used for signature verification.

API Request to retreive member details

In addition to the JWT received, you can also call userinfo endpoint to retrieve the member details.

HTTP

```
GET https://api.linkedin.com/v2/userinfo
Authorization: Bearer <access token>
```

Response Body Schema

Field Name	Description	Format
sub	Subject Identifier	Text
name	Full name	Text
given_name	Member's first name	Text
family_name	Member's last name	Text
picture	Member's profile picture URL	Text
locale	Member's locale	Text
email	Member's primary e-mail address.	Text
email_verified	Indicator that Member's primary email has been verified	Boolean

Sample API Response

JSON

```
{
    "sub": "782bbtaQ",
    "name": "John Doe",
    "given_name": "John",
    "family_name": "Doe",
    "picture": "https://media.licdn-
ei.com/dms/image/C5F03AQHqK8v7tB1HCQ/profile-displayphoto-
shrink_100_100/0/",
    "locale": "en-US",
    "email": "doe@email.com",
    "email_verified": true
}
```

With the member's profile information successfully retrieved, the sign-in process is now complete and your member can continue to enjoy their personalized experience with your site or application.

Additional Resources

Image Resources

Download official Sign In with LinkedIn button images in multiple sizes and formats from the LinkedIn Brand Resources archive for use within your site or application.

Rate Limits

Throttle Type	Daily Request Limit (UTC)
Member	500 Requests
Application	100,000 Requests

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Share on LinkedIn

Article • 12/14/2023

Overview

LinkedIn is a powerful platform to share content with your social network. Ensure your content receives the professional audience it deserves using Share on LinkedIn.

Use Share on LinkedIn to:

- Get your content in front of an audience of millions of professionals.
- Drive traffic to your site and grow your member base.
- Benefit from having your content shared across multiple professional networks worldwide.

Getting Started

Authenticating Members

New members Sharing on LinkedIn from your application for the first time will need to follow the Authenticating with OAuth 2.0 Guide. When requesting an authorization code in Step 2 of the OAuth 2.0 Guide, make sure to request the `w_member_social` scope!

Expand table

Permission Name	Description
w_member_social	Required to create a LinkedIn post on behalf of the authenticated member.

After successful authentication, you will acquire an access token that can be used in the next step of the share process.

If your application does not have this permission, you can add it through the Developer Portal. Select your app from My Apps, navigate to the Products tab, and add the Share on LinkedIn product which will grant you `w_member_social`.

Creating a Share on LinkedIn

There are multiple ways to share content with your LinkedIn network. In this guide, we will show you how to create shares using text, URLs, and images. For all shares created on LinkedIn, the request will always be a POST request to the User Generated Content (UGC) API.

API Request

HTTP

```
POST https://api.linkedin.com/v2/ugcPosts
```

> **Note**
>
> All requests require the following header: `X-Restli-Protocol-Version: 2.0.0`

Request Body Schema

Expand table

Field Name	Description	Format	Required
author	The author of a share contains Person URN of the Member creating the share. See Sign In with LinkedIn using OpenID Connect to see how to retrieve the Person URN.	Person URN	Yes
lifecycleState	Defines the state of the share. For the purposes of creating a share, the lifecycleState will always be `PUBLISHED`.	string	Yes
specificContent	Provides additional options while defining the content of the share.	ShareContent	Yes
visibility	Defines any visibility restrictions for the share. Possible values include: • `CONNECTIONS` - The share will be viewable by 1st-degree connections only. • `PUBLIC` - The share will be viewable by anyone on LinkedIn.	MemberNetworkVisibility	Yes

Share Content

Expand table

Field Name	Description	Format	Required
shareCommentary	Provides the primary content for the share.	string	Yes
shareMediaCategory	Represents the media assets attached to the share. Possible values include: • `NONE` - The share does not contain any media, and will only consist of text. • `ARTICLE` - The share contains a URL. • `IMAGE` - The Share contains an image.	string	Yes
media	If the shareMediaCategory is `ARTICLE` or `IMAGE`, define those media assets here.	ShareMedia[]	No

Share Media

Expand table

Field Name	Description	Format	Required
status	Must be configured to `READY`.	string	Yes
description	Provide a short description for your image or article.	string	No
media	ID of the uploaded image asset. If you are uploading an article, this field is not required.	DigitalMediaAsset URN	No
originalUrl	Provide the URL of the article you would like to share here.	string	No
title	Customize the title of your image or article.	string	No

Create a Text Share

The example below creates a simple text Share on LinkedIn. Notice the visibility is set to PUBLIC, where anyone on the LinkedIn Platform can view this share.

Sample Request Body

JSON

```
{
    "author": "urn:li:person:8675309",
    "lifecycleState": "PUBLISHED",
    "specificContent": {
        "com.linkedin.ugc.ShareContent": {
            "shareCommentary": {
                "text": "Hello World! This is my first Share on LinkedIn!"
            },
            "shareMediaCategory": "NONE"
        }
    },
    "visibility": {
        "com.linkedin.ugc.MemberNetworkVisibility": "PUBLIC"
    }
}
```

Response

A successful response will return `201 Created`, and the newly created post will be identified by the `X-RestLi-Id` response header.

Create an Article or URL Share

The example below illustrates various options when Sharing an Article or URL. The request body is similar to the Text Share above, however, we have now specified a media parameter containing the URL, title, and description. Keep in mind the title and description are optional parameters.

Sample Request Body

JSON

```
{
    "author": "urn:li:person:8675309",
    "lifecycleState": "PUBLISHED",
    "specificContent": {
        "com.linkedin.ugc.ShareContent": {
            "shareCommentary": {
                "text": "Learning more about LinkedIn by reading the
LinkedIn Blog!"
            },
            "shareMediaCategory": "ARTICLE",
            "media": [
                {
                    "status": "READY",
```

```
                    "description": {
                        "text": "Official LinkedIn Blog - Your source for
insights and information about LinkedIn."
                    },
                    "originalUrl": "https://blog.linkedin.com/",
                    "title": {
                        "text": "Official LinkedIn Blog"
                    }
                }
            ]
        }
    },
    "visibility": {
        "com.linkedin.ugc.MemberNetworkVisibility": "PUBLIC"
    }
}
```

Response

A successful response will return `201 Created`, and the newly created post will be identified by the `X-RestLi-Id` response header.

Create an Image or Video Share

If you'd like to attach an image or video to your share, you will first need to register, then upload your image/video to LinkedIn before the share can be created. We will walk through the following steps to create the share:

1. Register your image or video to be uploaded.
2. Upload your image or video to LinkedIn.
3. Create the image or video share.

Register the Image or Video

Send a POST request to the `assets` API, with the action query parameter to `registerUpload`.

HTTP

```
POST https://api.linkedin.com/v2/assets?action=registerUpload
```

Similar to the author parameter we've used with the ugcPosts API, we will need to provide our Person URN. Additional `recipes` and `serviceRelationships` define the type of content we're publishing. For Share on LinkedIn, recipes will always contain either the

type feedshare-image or the type feedshare-video (depending on which of the two you are uploading) and serviceRelationships will always define the relationshipType and identifier. See the request body below for reference.

JSON

```
{
    "registerUploadRequest": {
        "recipes": [
            "urn:li:digitalmediaRecipe:feedshare-image"
        ],
        "owner": "urn:li:person:8675309",
        "serviceRelationships": [
            {
                "relationshipType": "OWNER",
                "identifier": "urn:li:userGeneratedContent"
            }
        ]
    }
}
```

A successful response will contain an `uploadUrl` and `asset` that you will need to save for the next steps.

JSON

```
{
    "value": {
        "uploadMechanism": {
            "com.linkedin.digitalmedia.uploading.MediaUploadHttpRequest": {
                "headers": {},
                "uploadUrl":
"https://api.linkedin.com/mediaUpload/C5522AQGTYER3k3ByHQ/feedshare-
uploadedImage/0?
ca=vector_feedshare&cn=uploads&m=AQJbrN86Zm265gAAAWemyz2pxPSgONtBiZdchrgG872
QltnfYjnMdb2j3A&app=1953784&sync=0&v=beta&ut=2H-IhpbfXrRow1"
            }
        },
        "mediaArtifact": "urn:li:digitalmediaMediaArtifact:
(urn:li:digitalmediaAsset:C5522AQGTYER3k3ByHQ,urn:li:digitalmediaMediaArtifa
ctClass:feedshare-uploadedImage)",
        "asset": "urn:li:digitalmediaAsset:C5522AQGTYER3k3ByHQ"
    }
}
```

Upload Image or Video Binary File

Using the `uploadUrl` returned from Step 1, upload your image or video to LinkedIn. To upload your image or video, send a `POST` request to the `uploadUrl` with your image or video included as a binary file. The example below uses cURL to upload an image file.

Sample Request

Bash

```
curl -i --upload-file /Users/peter/Desktop/superneatimage.png --header
"Authorization: Bearer redacted"
'https://api.linkedin.com/mediaUpload/C5522AQGTYER3k3ByHQ/feedshare-
uploadedImage/0?
ca=vector_feedshare&cn=uploads&m=AQJbrN86Zm265gAAAWemyz2pxPSgONtBiZdchrgG872
QltnfYjnMdb2j3A&app=1953784&sync=0&v=beta&ut=2H-IhpbfXrRow1'
```

Create the Image or Video Share

After the image or video file has successfully uploaded from Step 2, we will use the `asset` from Step 1 to attach the image to our share. Below is a sample request for an image; for a video, the shareMediaCategory should be VIDEO instead of IMAGE.

Sample Request Body

JSON

```
{
    "author": "urn:li:person:8675309",
    "lifecycleState": "PUBLISHED",
    "specificContent": {
        "com.linkedin.ugc.ShareContent": {
            "shareCommentary": {
                "text": "Feeling inspired after meeting so many talented
individuals at this year's conference. #talentconnect"
            },
            "shareMediaCategory": "IMAGE",
            "media": [
                {
                    "status": "READY",
                    "description": {
                        "text": "Center stage!"
                    },
                    "media": "urn:li:digitalmediaAsset:C5422AQEbc381YmIuvg",
                    "title": {
                        "text": "LinkedIn Talent Connect 2021"
                    }
                }
            ]
```

```
        }
    },
    "visibility": {
        "com.linkedin.ugc.MemberNetworkVisibility": "PUBLIC"
    }
}
```

Response

A successful response will return `201 Created`, and the newly created post will be identified by the `X-RestLi-Id` response header.

Rate Limits

Expand table

Throttle Type	Daily Request Limit (UTC)
Member	150 Requests
Application	100,000 Requests

Feedback

Live Events APIs

Article • 05/08/2023

Overview

Use the Live Events API to stream and manage Live Events with your LinkedIn network. The Live Events API offers the following functionality:

1. Register the Live Event.
2. Ingest RTMP(s) content.
3. Create a Post to share your Live Event with your LinkedIn network.
4. End the Live Event.
5. Check the Live Event status.

Getting Started

Authenticating Members

New members authenticating with your developer application for the first time will need to follow the Authenticating with OAuth 2.0 Guide. When requesting an authorization code in Step 2 of the OAuth 2.0 Guide, make sure to request the minimum set of permission scopes required for your use case.

> **Note**
>
> Before authenticating with OAuth 2.0, determine if your use case permits creating live events for the member, or a company page. Combining both member and company page permission scopes within a single authentication request is not permitted.

> **Note**
>
> LinkedIn Live is currently a beta feature. As a broadcaster, to apply for access see **Applying for Live Events Broadcasting**.

Member Live Events

Permission Name	Description
`r_member_live`	READ access to a member's live events. This permission scope allows you to check the status of your Live Event, as well as retrieve the Live Event playable streams.
`w_member_live`	WRITE access to a member's live events. This permission scope allows you to upload and manage your Live Event.
`r_liteprofile`	READ access to a member's lite profile. This permission scope permits access to retrieve the user's ID via the Profile API.

Company Page Live Events

Permission Name	Description
`r_organization_live`	READ access to your organization's live events. This permission scope allows you to check the status of your Live Event, as well as retrieve the Live Event playable streams.
`w_organization_live`	WRITE access to your organization's live evends. This permission scope allows you to upload and manage your Live Events.
`r_organization_admin`	READ access to a member's organization or company page. This permission scope permits access to retrieve the organization ID via the Organization Access Control API.

Tiering

The Live Events API Program is available in two tiers: Development and Standard Tiers. The intent of the Development Tier is to introduce Developers to Live Events APIs, giving you limited access to create and view Live Events on LinkedIn. A prerequisite to requesting access to the Standard Tier will require a demonstration from your platform, showcasing each of the Live Events certification test cases. Access to both the Development and Standard Tiers are available through managing your developer applications Products at developer.linkedin.com.

Development Tier

The Developement Tier contains access to all APIs and services of the full Standard Tier, with a restriction on the total number of API calls your developer application can request within a 24 hour period. Generally, each Live Events API is limited to 100 API requests per day -- giving you the freedom to test and develop against LinkedIn APIs for

your developer application. The intent of the Development Tier is to give you a preview of the Live Events Program, and to demonstrate your proficiency in integrating with LinkedIn before being elevated to the Standard Tier.

Standard Tier

When you are ready to launch your LinkedIn Live Events integration to your broadcasters, you must request access to the Standard Tier. We will take this time to review your integration with LinkedIn, and require a demostration video showcasing your fulfillment of our Certification Test Cases.

Register

Register your Live Event using the `liveAssetActions` API with the `action` query parameter to `register`.

> ⓘ Note
>
> Newly registered Live Events will be discarded if ingestion has not started within 1 hour. After ingestion has started, a timeout will occur if the ingest URL has not received any data within 120 seconds.

Field Name	Type	Description
owner	Person or Organization URN	The unique identifier of the member or organization registering the asset. To identify your Person URN, use the Lite Profile API and append the ID returned to "urn:li:person:". To identify your Organization URN, use the Organization Access Control API. You must be the Administrator of your Organization in order to post to your Organization Page.
recipes	AssetRecipe	Defines the asset media type. The live event asset media type is always urn:li:digitalmediaRecipe:feedshare-live-video.

Field Name	Type	Description
region	string	The region specifies the closest region your asset should be registered to. Possible values include: 1. `WEST_US` (West US) 2. `EAST_US_NORTH` (Northeastern US) 3. `EAST_US_SOUTH` (Southeastern US) 4. `CENTRAL_US` (Central US) 5. `SOUTH_CENTRAL_US` (South Central US) 6. `SOUTH_AMERICA` (South America) 7. `NORTH_EUROPE` (North Europe) 8. `WEST_EUROPE` (West Europe)
autoCaptionLanguageTag	string (optional)	The BCP-47 language tag of the auto captions to generated for the live event. Use "en-US" for English. Remove this parameter if you do not want auto captions added.

Tip

Use **decoration syntax** to retrieve the Person ID within a single request. GET **https://api.linkedin.com/v2/me?projection=(id)**

Tip

Use **decoration syntax** to retrieve the Organization URN within a single request. **https://api.linkedin.com/v2/organizationAcls?q=roleAssignee&projection=(elements*(organization~))**

API Request

HTTP

```
POST https://api.linkedin.com/v2/liveAssetActions?action=register
```

Sample Request Body

JSON

```
{
    "registerLiveEventRequest": {
        "owner": "urn:li:person:12345",
```

```
        "recipes": ["urn:li:digitalmediaRecipe:feedshare-live-video"],
        "region": "WEST_US"
    }
}
```

A successful response will include an array of `ingestUrls` where RTMPS is the preferred ingestion protocol. Secondary ingest URLs are available in the event the primary ingest URL is unavailable.

Ingestion needs to begin before creating your Post on LinkedIn.

The `asset` ID will be used in subsequent requests to create a Post, end the Live Event.

Sample Response Body

JSON


```
{
    "value": {
        "ingestUrls": [
            {
                "ingestProtocol": "RTMP",
                "url":
"rtmp://12345.channel.media.azure.net:1935/live/12345"
            },
            {
                "ingestProtocol": "RTMP",
                "url":
"rtmp://12345.channel.media.azure.net:1936/live/12345"
            },
            {
                "ingestProtocol": "RTMPS",
                "url":
"rtmps://12345.channel.media.azure.net:2935/live/12345"
            },
            {
                "ingestProtocol": "RTMPS",
                "url":
"rtmps://12345.channel.media.azure.net:2936/live/12345"
            }
        ],
        "previewUrls": [
            "https://12345-vectoreimedia2.preview-
usw22.channel.media.azure.net/e57e889f-aaaa-bbbb-cccc-
c337afacf31a/preview.ism/manifest"
        ],
        "mediaArtifact": "urn:li:digitalmediaMediaArtifact:
(urn:li:digitalmediaAsset:12345,urn:li:digitalmediaMediaArtifactClass:feedsh
are-live-liveinput)",
        "asset": "urn:li:digitalmediaAsset:12345"
```

```
    }
}
```

> ⓘ Note
>
> We recommend using RTMPS whenever available. If you would like to take advantage of a backup stream, a separate time-synced hardware encoder can be used to stream to the secondary RTMP(s) urls provided.

Live Ingest Requirements

Use the following specifications when encoding your Live Event:

Field Name	Description
Duration	max 4 hours. Live Events may not exceed 4 hour limit.
Aspect Ratio	16:9
Resolution	max 1080p
Frame Rate	max 30 fps
Key Frame	every 2 seconds (60 frames)
Bitrate	max 6 mbps video; max 128 kbps audio, 48 khz sample rate
Encoding	H264 video, AAC audio
Protocol	RTMP/RTMPS (preferred)

> ⓘ Note
>
> When streaming via RTMP, check firewall and/or proxy settings to confirm that outbound TCP ports 1935 and 1936 are open. When streaming via RTMPS, check firewall and/or proxy settings to confirm that outbound TCP ports 2935 and 2936 are open. For a complete list of IP ranges required for allowlisting, please refer to **Microsoft Azure IP Ranges**.

Ingest and Create a User Generated Content (UGC) Post

Using the RTMP or RTMPS ingest URLs from the previous step, you may now begin ingestion of your Live Event.

Before creating the Post, first check if the Live ViEventdeo is ready to be shared on LinkedIn.

Check Recipe Status

To check the status of your Live Event, send a GET request to the `assets` API. After successful ingestion, the recipe status is `AVAILABLE`, and ready to be included with your UGCPost.

> ⓘ **Note**
>
> If your asset recipe status has not updated after 15 seconds, there may be an issue with ingestion. End the current session by sending the action to endLiveEvent, and try again by registering a new Live Event.

Recipe Status	Description
NEW	A newly assigned recipe. This status is only used for new assignments. Transitions to PROCESSING as soon as processing requests are sent.
PROCESSING	Some or all of the recipe's required artifacts are not available and processing is underway to generate the missing artifacts.
AVAILABLE	All of the recipe's required artifacts are available.
INCOMPLETE	The artifact is not available because it has been deleted or is in deleting process.

API Request

HTTP

```
GET https://api.linkedin.com/v2/assets/:id
```

Sample API Response

JSON

```
{
    "owner": "urn:li:person:12345",
    "resourceRelationships": [],
    "serviceRelationships": [],
    "recipes": [
        {
            "recipe": "urn:li:digitalmediaRecipe:feedshare-live-video",
            "status": "AVAILABLE"
        }
    ],
    "mediaTypeFamily": "VIDEO",
    "created": 1539649231180,
    "executedActions": [
        {
            "recipe": "urn:li:digitalmediaRecipe:feedshare-live-video",
            "action": "PROCESSING",
            "createdAt": 1539649231234
        },
        {
            "recipe": "urn:li:digitalmediaRecipe:feedshare-live-video",
            "action": "PROCESSED",
            "createdAt": 1539649504293
        }
    ],
    "lastModified": 1539649504293,
    "id": "12345",
    "status": "ALLOWED"
}
```


Use the `ugcPosts` API to create a Post on LinkedIn, viewable by your LinkedIn network. Include the asset ID returned from registering the Live Event, as well as relevant Post metadata in the request body. Keep in mind that the UGC Post must be created as soon as you begin ingestion.

API Request

HTTP

```
POST https://api.linkedin.com/v2/ugcPosts
```

> ⓘ **Note**
>
> All requests require the following header: `X-Restli-Protocol-Version: 2.0.0`

Request Body Schema

Field Name	Description	Format	Required
author	The author of a share contains either the Person or Organization URN.	Person URN Organization URN	Yes
lifecycleState	Defines the state of the share. For the purposes of creating a share, the lifecycleState will always be `PUBLISHED`.	string	Yes
specificContent	Provides additional options while defining the content of the share.	ShareContent	Yes
visibility	Defines any visibility restrictions for the share. Possible values include: • `PUBLIC` - The share will be viewable by anyone on LinkedIn.	MemberNetworkVisibility	Yes

Share Content

Field Name	Description	Format	Required
shareCommentary	Provides the primary content for the share.	string	Yes
shareMediaCategory	Represents the media assets attached to the share. Possible values include: • `LIVE_VIDEO` - The share contains live event content.	string	Yes
media	Use the Asset URN returned from registering the live event.	ShareMedia[]	No

Share Media

Field Name	Description	Format	Required
status	Must be configured to `READY`.	string	Yes
media	ID of the uploaded image asset. If you are uploading an article, this field is not required.	DigitalMediaAsset URN	No

Request Headers

Header	Value

Header	Value
X-RestLi-Method	Create

Sample Request Body

JSON

```
{
    "author": "urn:li:person:abcde12345",
    "lifecycleState": "PUBLISHED",
    "specificContent": {
        "com.linkedin.ugc.ShareContent": {
            "media": [
                {
                    "media": "urn:li:digitalmediaAsset:12345",
                    "status": "READY"
                }
            ],
            "shareCommentary": {
                "attributes": [],
                "text": "Join us as we live stream!"
            },
            "shareMediaCategory": "LIVE_VIDEO"
        }
    },
    "visibility": {
        "com.linkedin.ugc.MemberNetworkVisibility": "PUBLIC"
    }
}
```

A 201 response confirms your UGC Post has been created successfully. To generate a user-friendly URL to your newly created UGC Post, identify the **X-RestLi-Id** within the response header. Append the header value to `https://linkedin.com/feed/update/`. The resulting URL should resemble:

HTTP

```
https://www.linkedin.com/video/live/urn:li:ugcPost:1238957139875
```

End the Live Event

Once your Live Event has ended, send an `action` to `end` for your asset ID.

API Request

HTTP

```
POST https://api.linkedin.com/v2/liveAssetActions?action=end
```

Sample Request Body

JSON

```
{
    "asset": "urn:li:digitalmediaAsset:C5624AQEUbk4_xZgHJQ"
}
```

> **Note**
>
> We recommend waiting 10 seconds after your Live Event has ended before submitting the action to endLiveEvent to ensure your broadcast is not interrupted.

A successful response is indicated by a `200` Response Code.

Feedback

Live Events Content Access API

Article • 05/08/2023

Overview

LinkedIn Live provides a platform for individuals and organizations to broadcast live event content to their network in real time. This video broadcasting feature is currently in beta and available to LinkedIn Members and LinkedIn Pages that meet specific criteria. This specific criteria is to ensure we provide access to existing content creators who have a significant LinkedIn audience and a history of creating quality, professional video content. We also want to ensure that video content sharing happens in a safe and trusted environment on LinkedIn.

To learn more about requesting access to LinkedIn Live, submit an application here

Already have access? Use this API to confirm whether your profile or page have been approved. The response from this will API will determine if you have access to create live videos using the following services: assets, liveVideos, and ugcPosts.

Content Access

API Request

HTTP

```
GET https://api.linkedin.com/v2/contentAccess/
```

Expand table

Key Name	Type	Description	Required
entity: (member)	Person URN	Used when requesting content access information on behalf of a LinkedIn Member (Profile). Accepts the URL-encoded person urn of the member going live.	Yes (Profile)
entity: (company)	Organization URN	Used when requesting content access information on behalf of a LinkedIn Page. Accepts the URL-encoded organization urn.	Yes (Pages)

Key Name	Type	Description	Required
admin: (member)	Person URN	Used when requesting content access information on behalf of a LinkedIn Page. Accepts the URL-encoded person urn of the page admin going live.	Yes (Pages)
featureType	string	Must always be the value: `LIVE_VIDEO`.	Yes

> **Note**
>
> You must include the `X-RestLi-Protocol-Version: 2.0.0` header along with your request.

Sample Request (Profile)

HTTP

```
GET https://api.linkedin.com/v2/contentAccess/(entity:
(member:urn%3Ali%3Aperson%3AmvgmS_tF6N),featureType:LIVE_VIDEO)
```

In this request, we append the URL-encoded person urn to the entity:(member) parameter.

URL-decoded person urn: `urn:li:person:mvgmS_tF6N` URL-encoded person urn: `urn%3Ali%3Aperson%3AmvgmS_tF6N` Entity: `entity:(member:urn%3Ali%3Aperson%3AmvgmS_tF6N)`

Sample Request (Page)

HTTP

```
GET 'https://api.linkedin.com/v2/contentAccess/(entity:
(company:urn%3Ali%3Aorganization%3A20277203),admin:
(member:urn%3Ali%3Aperson%3AmvgmS_tF6N),featureType:LIVE_VIDEO)
```

In this request, we append the URL-encoded organization urn to the entity:(company) parameter. We also append the URL-encoded person urn to the admin:(member) parameter.

URL-decoded organization urn: `urn:li:organization:20277203` URL-encoded organization urn: `urn%3Ali%3Aorganization%3A20277203` Entity: `entity:(company:urn%3Ali%3Aorganization%3A20277203)`

URL-decoded person urn: `urn:li:person:mvgmS_tF6N` URL-encoded person urn: `urn%3Ali%3Aperson%3AmvgmS_tF6N` Admin: `admin:(member:urn%3Ali%3Aperson%3AmvgmS_tF6N)`

API Response

If the member or page has been approved to go live, the API returns a `200 OK` response.

If the member or page has not yet been approved to go live, the API returns a `404 Not Found` response.

Feedback

Scheduled Live Events

Article • 05/08/2023

Overall Flow

You can schedule your Page or profile's scheduled live event in advance, and then promote it on and off LinkedIn. The flow to create a scheduled live event is outlined below.

> **Note**
>
> Members are currently limited to scheduling up to 10 scheduled live events per day (including both your profile and any Pages you manage).

1. If you would like to include an image with a scheduled live event announcement post, upload an announcement image `asset`. (optional)

```
https

POST /assets?action=registerUpload
```

2. Create a scheduled `liveVideo` asset. Use the `asset` from step #1 if available.

```
https

POST /liveVideos
```

3. Create an annoucement post. Use the `liveVideo` urn from step #2.

```
https

POST /ugcPosts
```

4. When you are ready to go live and are nearing your scheduled live event time, register the live event `asset`.

> **Note**
>
> The allowed go live time window is between 15 minutes before the scheduled time and 2 hours after the scheduled time.

https

```
POST /liveAssetActions?action=register
```

5. Begin RTMP(s) ingestion.
6. Update the scheduled `liveVideo` (step #2) with the live event `asset` from step #4. This step can only be completed once the `asset` status is AVAILABLE.

https

```
POST /liveVideos
```

7. End the scheduled live event.

https

```
POST https://api.linkedin.com/v2/liveAssetActions?action=end
```

1. Upload an Announcement Image (optional)

Upload an announcement image to increase engagement with your scheduled live event. This step is optional. If no announcement image is provided, a default announcement image will be used with the announcement post.

Supported Input Format: JPEG, PNG

Image Requirements:

- **preferred aspect ratio**: 16:9
- **max pixel limit (w*h)**: 36152320
- **max image size**: 209mb

The first request will register your announcement image asset. As with all Live Events APIs, the owner provided must be the author (person or organization URN) of the live event. Save the announcement image asset urn to be used in later steps.

Use the `uploadUrl` returned from this request to upload the announcement image.

Request

JSON

```
POST /assets?action=registerUpload

curl -X POST 'https://api.linkedin.com/v2/assets?action=registerUpload' -H
'X-RestLi-Protocol-Version: 2.0.0' -H 'Content-Type: application/json' -H
'Authorization: Bearer <redacted>' --data-raw '{
  "registerUploadRequest": {
    "recipes": [
      "urn:li:digitalmediaRecipe:video-liveannouncement-image"
    ],
    "owner": "urn:li:person:iKO_ntS9UQ",
    "serviceRelationships": [
      {
        "relationshipType": "OWNER",
        "identifier": "urn:li:userGeneratedContent"
      }
    ]
  }
}'
```

Response

JSON


```
{
    "value": {
        "uploadMechanism": {
            "com.linkedin.digitalmedia.uploading.MediaUploadHttpRequest": {
                "uploadUrl":
"https://api.linkedin.com/mediaUpload/C5F22AQEUvl4IOBjE2g/feedshare-
uploadedImage/0?
ca=vector_feedshare&cn=uploads&m=AQJ2Wjp_VaTk1wAAAXKfCVcFPtCwi6mquOfiHR356tr
Dm59sEEg77qwSpA&app=1953784&sync=0&v=beta&ut=1hrN_TF9ERQpg1",
                "headers": {
                    "media-type-family": "STILLIMAGE"
                }
            }
        },
        "asset": "urn:li:digitalmediaAsset:C5F22AQEUvl4IOBjE2g",
        "mediaArtifact": "urn:li:digitalmediaMediaArtifact:
(urn:li:digitalmediaAsset:C5F22AQEUvl4IOBjE2g,urn:li:digitalmediaMediaArtifa
ctClass:feedshare-uploadedImage)"
    }
}
```


Uploading the image using the uploadUrl in a curl request

curl

```
curl --upload-file ~/Downloads/liveAnnouncement.jpg -H 'Authorization:
Bearer <redacted>'
'https://api.linkedin.com/mediaUpload/C5F22AQEUvl4IOBjE2g/feedshare-
uploadedImage/0?
ca=vector_feedshare&cn=uploads&m=AQJ2Wjp_VaTk1wAAAXKfCVcFPtCwi6mquOfiHR356tr
Dm59sEEg77qwSpA&app=1953784&sync=0&v=beta&ut=1hrN_TF9ERQpg1'
```

2. Create a Scheduled Live Event Asset

The scheduledAt parameter defines the epoch time in milliseconds of your scheduled live event.

For example, a scheduledAt time of `1588550400000` represents Monday, May 4, 2020 12:00:00 AM GMT.

The `name` parameter must be used to provide a title for your scheduled live event.

> ⓘ Note
>
> The `name` has a max length limit of 75 chars.

If an `announcementImage` is available, include the asset urn from step #1, and provide an image title. If there is no announcement image, remove the announcementImage parameter from the request body.

(Member Profile) Request

JSON

```
POST /liveVideos

curl -i -X POST 'https://api.linkedin.com/v2/liveVideos' -H 'X-RestLi-
Protocol-Version: 2.0.0' -H "Content-Type: application/json" -H
"Authorization: Bearer <redacted>" --data-raw '{
  "author": {
    "member": "urn:li:person:iKO_ntS9UQ"
  },
  "scheduledAt": 1591892529000,
  "announcementImage": {
    "media": "urn:li:digitalmediaAsset:C5F22AQEUvl4IOBjE2g",
    "title": {
      "text": "Title of the image.",
      "inferredLocale": "en_US"
    }
  },
```

```
  "name": "Title of the scheduled live event"
}'
```

(Organization) Request

JSON

```
POST /liveVideos

curl -i -X POST 'https://api.linkedin.com/v2/liveVideos' -H 'X-RestLi-
Protocol-Version: 2.0.0' -H "Content-Type: application/json" -H
"Authorization: Bearer <redacted>" --data-raw '{
  "author": {
    "organization": "urn:li:organization:0000"
  },
  "scheduledAt": 1591892529000,
  "announcementImage": {
    "media": "urn:li:digitalmediaAsset:C5F22AQEUvl4IOBjE2g",
    "title": {
      "text": "Title of the image.",
      "inferredLocale": "en_US"
    }
  },
  "name": "Title of the scheduled live event"
}'
```

Response

Save the location header id returned within the response. Appending this id to `urn:li:liveVideo` will be used when creating the announcement post.

JSON

```
HTTP/2 201
x-li-responseorigin: RGW
location: /liveVideos/6676519335839784960
x-restli-id: 6676519335839784960
x-restli-protocol-version: 2.0.0
```

3. Create an Announcement Post

Creating the announcement post that will publish to your LinkedIn feed is similar to the existing live event ugc post, with a few key differences. Specifically, the `media`, `shareMediaCategory`, and `distribution` parameters will need to be updated for scheduled posts.

The `media` parameter must include the liveVideo urn (`urn:li:liveVideo:6676519335839784960`) saved from step #2.

The `shareMediaCategory` is now defined as `URN_REFERENCE`.

The `distribution` parameters are defined in the sample request below.

Request

JSON

```
POST /ugcPosts

curl -X POST 'https://api.linkedin.com/v2/ugcPosts' -H 'X-RestLi-Protocol-
Version: 2.0.0' -H "Content-Type: application/json" -H "Authorization:
Bearer <redacted>" --data-raw '{
  "author": "urn:li:person:iKO_ntS9UQ",
  "lifecycleState": "PUBLISHED",
  "specificContent": {
    "com.linkedin.ugc.ShareContent": {
      "media": [
        {
          "media": "urn:li:liveVideo:6676519335839784960",
          "status": "READY"
        }
      ],
      "shareCommentary": {
        "attributes": [],
        "text": "This is the text description displayed with the post."
      },
      "shareMediaCategory": "URN_REFERENCE"
    }
  },
  "visibility": {
    "com.linkedin.ugc.MemberNetworkVisibility": "PUBLIC"
  },
  "distribution": {
    "feedDistribution": "MAIN_FEED",
    "externalDistributionChannels": [],
    "distributedViaFollowFeed": true
  }
}'
```

Response

Appending the id returned to `https://www.linkedin.com/feed/update/urn:li:ugcPost:6676519700211556352` provides a reference for users to view the scheduled post on LinkedIn.

JSON

```
{
    "id": "urn:li:ugcPost:6676519700211556352"
}
```

4. Register the Live Event

When you are ready to go live to your scheduled live event, begin the process to register your asset and begin RTMP ingestion. This step remains unchanged from the existing live event asset registration. For reference, use the request below.

> ⓘ **Note**
>
> Newly registered Live Events will be discarded if ingestion has not started within 4 hours. After ingestion has started, a timeout will occur if the ingest URL has not received any data within 90 seconds.

Request

JSON

```
POST https://api.linkedin.com/v2/liveAssetActions?action=register

curl -X POST 'https://api.linkedin.com/v2/liveAssetActions?action=register'
\
-H 'X-RestLi-Protocol-Version: 2.0.0' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <redacted>' \
--data-raw '{
    "registerLiveEventRequest": {
        "owner": "urn:li:person:iKO_ntS9UQ",
        "recipes": ["urn:li:digitalmediaRecipe:feedshare-live-video"],
        "region": "WEST_US"
    }
}'
```

5. Begin RTMP(s) Ingestion

RTMP(s) ingestion may only begin 15 minutes prior to your `scheduledAt`, or 2 hours after your `scheduledAt` time.

6. Update the Scheduled Live Event

Using the registered live event asset from step #4, we can update the scheduled live event (step #2) and effectively transition our announcement post to a scheduled live event. You will need the live event id saved from step #2 to update the `liveVideoAsset` parameter. Ensure the live event asset status is AVAILABLE before proceeding with this step.

A successful response returns a 204 that your asset has been updated.

Request

JSON

```
POST /liveVideos

curl -X POST 'https://api.linkedin.com/v2/liveVideos/6676519335839784960' -H
'X-RestLi-Protocol-Version: 2.0.0' -H "Content-Type: application/json" -H
"Authorization: Bearer <redacted>" --data-raw '{
  "patch": {
    "$set": {
      "liveVideoAsset": {
        "media": "urn:li:digitalmediaAsset:C5524AQFazrOYw4_q2Q"
      }
    }
  }
}'
```

7. End Live Event Asset

For any live Event asset created, once ingestion has completed always be sure to conclude your event by sending the action to end live event. This step remains unchanged. For reference, use the following request:

Request

JSON

```
POST https://api.linkedin.com/v2/liveAssetActions?action=end

curl -X POST 'https://api.linkedin.com/v2/liveAssetActions?action=end' -H
"Authorization: Bearer <redacted>" --data-raw '{
  {
      "asset": "urn:li:digitalmediaAsset:C5624AQEUbk4_xZgHJQ"
```

```
  }
}'
```

Retrieving Information about Created Scheduled Live Events

To retrieve information about all scheduled live events by the author, send a GET request to the live events API. You must include the encoded author urn along with your request. Possible values for the author urn include: person urn or organization urn.

Request

JSON

```
curl -X GET 'https://api.linkedin.com/v2/liveVideos?
q=author&author=urn%3Ali%3Aperson%3AiKO_ntS9UQ' -H 'X-RestLi-Protocol-
Version: 2.0.0' -H "Content-Type: application/json" -H "Authorization:
Bearer <redacted>"
```

Response

JSON


```
{
    "paging": {
        "start": 0,
        "count": 15,
        "links": [],
        "total": 4
    },
    "elements": [
        {
            "scheduledAt": 1591892529000,
            "name": "Title Goes West",
            "liveVideoAsset": {
                "media": "urn:li:digitalmediaAsset:C5524AQFazrOYw4_q2Q"
            },
            "id": 6676519335839784960
        },
        {
            "scheduledAt": 1591744521000,
            "id": 6676245139628879872
        },
        {
            "scheduledAt": 1591731035000,
            "id": 6676183242384732160
```

```
        },
        {
            "scheduledAt": 1592004112000,
            "id": 6676174265252966400
        }
    ]
}
```

To retrieve information about a single scheduled live event, you may send a GET request to the liveVideos service along with the live events id.

Request

JSON

```
curl -X GET 'https://api.linkedin.com/v2/liveVideos/6676519335839784960' -H
'X-RestLi-Protocol-Version: 2.0.0' -H "Content-Type: application/json" -H
"Authorization: Bearer <redacted>"
```

Response

JSON

```
{
    "scheduledAt": 1591892529000,
    "name": "Title Goes West",
    "liveVideoAsset": {
        "media": "urn:li:digitalmediaAsset:C5524AQFazrOYw4_q2Q"
    },
    "id": 6676519335839784960
}
```

Feedback

Live Events Target Audiences

Article • 05/08/2023

With Target Audiences, you can now specify a country, region, or language along with your Organization's Live Events posts. Target Audiences is available for use with Company Pages only. To implement target audiences, there will be a few key changes to your user workflow.

1. Register the Live Event.
2. Ingest RTMP(s) content.
3. *NEW* Identify countries, regions, and languages to target. Use the geoTypeahead service to allow members to search for countries or regions. Refer to ISO-639 for a list of supported langauges.
4. *NEW* Create a Post with a list of countries, regions, and languages to target to share your Live Event with your LinkedIn network.
5. End the Live Event.

Geo Typeahead

The Bing Geo Typeahead API will provide a list of corresponding Geo URNs given a user input search query. The display name represents the user-friendly representation of the country or region, while the entity represents the URN that will be appended to the ugcPost.

> **Note**
>
> All use of the Microsoft Bing Maps location data is subject to **Microsoft Bing Maps and MapPoint Web Service End User Terms of Use and Embedded Maps Service Terms of Use** and the **Microsoft Privacy Statement**. By accessing any Microsoft Bing Maps location data, you are agreeing to be bound by these Microsoft terms.

Field	Sub-Field	Description	Format
hits		The record containing the typeahead hit information.	Array
	displayName	Hit title. The display name of the geo URN.	String
	entity	The urn matching the search query.	URN

API Request

HTTP

```
GET https://api.linkedin.com/v2/geoTypeahead?q=search&query=San
GET https://api.linkedin.com/v2/geoTypeahead?q=search&query=San&locale=
(language:en,country:US)
```

Parameters

Field Name	Sub-Field Name	Description	Required
query		Typeahead query used to find available geo results given the user input.	Yes
locale		The locale the country data is requested in, as represented by the two-letter language and country codes. Defaults to `en_US`.	Optional
	language	A lowercase two-letter language code as defined by ISO-639.	Optional
	country	An uppercase two-letter country code as defined by ISO-3166.	Optional

Sample Response Body

JSON

```
{
    "paging": {
        "start": 0,
        "count": 10,
        "links": [
            {
                "type": "application/json",
                "rel": "next",
                "href": "/v2/geoTypeahead?
q=search&query=san&start=10&count=10&locale=(country:US,language:en)"
            }
        ],
        "total": 0
    },
    "elements": [
        {
            "displayText": "San Francisco Bay Area",
            "entity": "urn:li:geo:90000084"
        },
        {
            "displayText": "San Diego Metropolitan Area",
            "entity": "urn:li:geo:90010472"
        },
```

```
        {
            "displayText": "San Antonio, Texas Metropolitan Area",
            "entity": "urn:li:geo:90000724"
        },
        {
            "displayText": "Santiago Metropolitan Area",
            "entity": "urn:li:geo:90009899"
        },
        {
            "displayText": "San Juan-Carolina Area",
            "entity": "urn:li:geo:90009445"
        },
        {
            "displayText": "San Luis Potosí Metropolitan Area",
            "entity": "urn:li:geo:90010051"
        },
        {
            "displayText": "San José Metropolitan Area",
            "entity": "urn:li:geo:90010358"
        },
        {
            "displayText": "Santa Barbara-Santa Maria Area",
            "entity": "urn:li:geo:90010474"
        },
        {
            "displayText": "San Cristóbal Metropolitan Area",
            "entity": "urn:li:geo:90010453"
        },
        {
            "displayText": "San Angelo Area",
            "entity": "urn:li:geo:90000720"
        }
    ]
}
```


Locales

The `interfaceLocales` object represents an array of Locales to target. You can specify locales using the format below:

Schema

Field Name	Description	Required
language	A lowercase two-letter language code as defined by ISO-639.	Optional

Sample JSON

JSON

```
{
  "interfaceLocales": [
    {
      "language": "en"
    },
    {
      "language": "es"
    }
  ]
```

Audience Counts

With the Audience Counts API, you can forecast the reach of your target audience. Use the Audience Counts API to ensure your target audience contains at least 300 followers.

Schema

Field Name	Description
active	Active audience count for the given targeting criteria. Active audience includes members that are more likely to visit LinkedIn sites.
total	Total audience count for the given targeting criteria. To protect member privacy, this value is 0 when the audience size is less than 300. The total is a rounded approximation, and the degree of approximation depends upon the size of the audience segment.

The following TargetingCriteria can be used with your request. See here for more information about Targeting Criteria.

TargetingCriteria	Description
urn:li:adTargetingFacet:followedCompanies	include the Organization URN (urn:li:organization:3487269) here
urn:li:adTargetingFacet:interfaceLocales	include the list of Locales (urn:li:locale:en_US) here
urn:li:adTargetingFacet:locations	include the list of Location URNs (urn:li:geo:90000084) here

All API requests require the header: `X-Restli-Protocol-Version: 2.0.0`.

To comply with Restli 2.0 format, all URNs must be URL encoded. For example, `urn:li:skill:17` is URL encoded as `urn%3Ali%3Askill%3A17`.

> ⓘ Note
>
> Postman or similar API tools may not support URL encoding query parameters within the request. If you encounter the response: `Invalid query parameters passed to request`, please test your request using `curl`.

API Request

This sample submits a request to the `audienceCountsV2` service to find the audience size of Organization 5340951 under the Geo Location 90000084 and Interface Locale en_US.

HTTP

```
GET https://api.linkedin.com/v2/audienceCountsV2?
q=targetingCriteriaV2&targetingCriteria=(include:(and:List((or:
(urn%3Ali%3AadTargetingFacet%3AfollowedCompanies:List(urn%3Ali%3Aorganizatio
n%3A5340951))),(or:
(urn%3Ali%3AadTargetingFacet%3Alocations:List(urn%3Ali%3Ageo%3A90000084))),
(or:
(urn%3Ali%3AadTargetingFacet%3AinterfaceLocales:List(urn%3Ali%3Alocale%3Aen_
US))))))'
```

Parameters

Field Name	Required	Type	Description
q	This field should always be `targetingCriteriaV2`	string	Yes
targetingCriteria	Yes	targetingCriteria object	Specifies the targeting criteria that the member should match.

For reference, the unencoded and encoded Targeting Criteria used in the sample request is provided below.

Encoded Targeting Criteria

HTTP

```
=(include:(and:List((or:
(urn%3Ali%3AadTargetingFacet%3AfollowedCompanies:List(urn%3Ali%3Aorganizatio
n%3A5340951))),(or:
```

```
(urn%3Ali%3AadTargetingFacet%3Alocations:List(urn%3Ali%3Ageo%3A90000084))),
(or:
(urn%3Ali%3AadTargetingFacet%3AinterfaceLocales:List(urn%3Ali%3Alocale%3Aen_
US))))))
```

Unencoded Targeting Criteria

HTTP

```
=(include:(and:List((or:
(urn:li:adTargetingFacet:followedCompanies:List(urn:li:organization:5340951)
)),(or:(urn:li:adTargetingFacet:locations:List(urn:li:geo:90000084))),(or:
(urn:li:adTargetingFacet:interfaceLocales:List(urn:li:locale:en_US))))))
```

JSON Representation

JSON

```
{
  "targetingCriteria": {
    "include": {
      "and": [
        {
          "or": {
            "urn:li:adTargetingFacet:followedCompanies": [
              "urn:li:organization:5340951"
            ]
          }
        },
        {
          "or": {
            "urn:li:adTargetingFacet:interfaceLocales": [
              "urn:li:locale:en_US"
            ],
            "urn:li:adTargetingFacet:locations": [
              "urn:li:geo:90000084"
            ]
          }
        }
      ]
    }
  }
}
```

Sample Response

JSON

```
{
  "elements": [
    {
      "total": 420,
      "active": 0
    }
  ],
  "paging": {
    "count": 10,
    "start": 0,
    "links": []
  }
}
```


ugcPosts

API Request

HTTP

```
POST https://api.linkedin.com/v2/ugcPosts
```

> ⓘ Note
>
> All requests require the following header: `X-Restli-Protocol-Version: 2.0.0`

Request Body Schema

Field Name	Description	Format	Required
author	The author of a share contains either the Person or Organization URN.	Person URN Organization URN	Yes
lifecycleState	Defines the state of the share. For the purposes of creating a share, the lifecycleState will always be `PUBLISHED`.	string	Yes
specificContent	Provides additional options while defining the content of the share.	ShareContent	Yes

Field Name	Description	Format	Required
visibility	Defines any visibility restrictions for the share. Possible values include: • `PUBLIC` - The share will be viewable by anyone on LinkedIn.	MemberNetworkVisibility	Yes
targetAudience	Intended audience or best fit audiences for this content as decided by the owner.	TargetAudience	Optional

Share Content

Field Name	Description	Format	Required
shareCommentary	Provides the primary content for the share.	string	Yes
shareMediaCategory	Represents the media assets attached to the share. Possible values include: • `LIVE_VIDEO` - The share contains live event content.	string	Yes
media	Use the Asset URN returned from registering the live event.	ShareMedia[]	No

Share Media

Field Name	Description	Format	Required
status	Must be configured to `READY`.	string	Yes
media	ID of the uploaded image asset. If you are uploading an article, this field is not required.	DigitalMediaAsset URN	No

TargetAudience

Field	Sub-Field	Description	Format
targetedEntities		The entities targeted for distribution. Structured as an object containing multiple arrays of targeting entity URNs that functions like a series of OR conditionals.	Array of Targets (see below)

Field	Sub-Field	Description	Format
	geoLocations	Array of countries and regions to target. Countries and regions are represented as a Geo URN.	Array of Geo URN
	interfaceLocales	Array of languages to target. Languages are represented as a Locale.	Array of Locale

> ⓘ Note
>
> Your target audience must include at least 300 members.

Request Headers

Header	Value
X-RestLi-Method	Create

Sample Request Body

JSON

```
{
   "author":"urn:li:organization:abcde12345",
   "lifecycleState":"PUBLISHED",
   "specificContent":{
      "com.linkedin.ugc.ShareContent":{
         "media":[
            {
               "media":"urn:li:digitalmediaAsset:12345",
               "status":"READY"
            }
         ],
         "shareCommentary":{
            "attributes":[

            ],
            "text":"Join us as we live stream!"
         },
         "shareMediaCategory":"LIVE_VIDEO"
      }
   },
   "visibility":{
      "com.linkedin.ugc.MemberNetworkVisibility":"PUBLIC"
   },
```

```
   "targetAudience":{
      "targetedEntities":[
         {
            "geoLocations":[
               "urn:li:geo:867",
               "urn:li:geo:5309"
            ],
            "interfaceLocales":[
               {
                  "language":"en"
               },
               {
                  "language":"es"
               }
            ]
         }
      ]
   }
}
```

A 201 response confirms your UGC Post has been created successfully. To generate a user-friendly URL to your newly created UGC Post, identify the **X-RestLi-Id** within the response header. Append the header value to `https://linkedin.com/video/live/`. The resulting URL should resemble:

HTTP

```
https://www.linkedin.com/video/live/urn:li:ugcPost:1238957139875
```

Feedback

Live Events Announcements & Release Notes

Article • 06/30/2023

Bookmark this page to get the latest Live Events API announcements and release notes.

June 30, 2023

Update to the Available Asset Regions

After reviewing the activity for each live asset region, we have made the decision to remove two regions from the rotation. We have updated the asset region list to remove both SOUTHEAST_ASIA and CENTRAL_INDIA. The updated list of available regions can be found below.

1. `WEST_US` (West US)
2. `EAST_US_NORTH` (Northeastern US)
3. `EAST_US_SOUTH` (Southeastern US)
4. `CENTRAL_US` (Central US)
5. `SOUTH_CENTRAL_US` (South Central US)
6. `SOUTH_AMERICA` (South America)
7. `NORTH_EUROPE` (North Europe)
8. `WEST_EUROPE` (West Europe)

Oct 11, 2022

Introducing the Live Events Development Tier

We are proud to introduce the Live Events API Program to a broader audience. Starting today, all partners may request access to Live Events APIs through developer.linkedin.com .

Sept. 3, 2021

Migration from /organizationalEntityAcls to /organizationAcls

In order to go live from a Page, you first need to identify the Page's organization urn. The `organizationalEntityAcls` service has been marked for deprecation by November 2021, and will be replaced with `organizationAcls`. Follow the steps below to migrate your apps to use the new service.

organizationalEntityAcls Request

curl

```
https://api.linkedin.com/v2/organizationalEntityAcls?
q=roleAssignee&role=ADMINISTRATOR&projection=(elements*
(organizationalTarget~))
```

organizationAcls Request

curl

```
https://api.linkedin.com/v2/organizationAcls?q=roleAssignee&projection=
(elements*(organization~))
```

Scheduled Live Events Updates

We've clarified a few points in the Scheduled Live Events documentation where we've seen recent issues crop up.

1. The `name` or title of the scheduled live event has a max character limit of 75 chars.
2. We've added a note that live event `assets` can only be used once their state has transitioned to the AVAILABLE state.

July 1, 2021

Migration from /assets to /liveAssetActions

In efforts to improve performance and stability of live event broadcasts, we are introducing a new /liveAssetActions API to handle all requests to register, and end live events. To get started with the new service, you will need to make a few changes to your existing requests to /assets. The /liveAssetActions API will only serve to replace requests to register and end live events. There are no changes to retrieve asset status.

1. Update the resource name from `/assets` to `/liveAssetActions`.

2. Update `?action=registerLiveEvent` to `?action=register`. No changes to the request or response bodies.
3. Update `?action=endLiveEvent` to `?action=end`. The asset URN must now be provided in the request body. See End the Live Event for reference.

Typeahead for Bing Geo

- GeoTypeahead API '/geoTypeahead?q=federated' will be deprecated on **August 31, 2021**. Please use GeoTypeahead API '/geoTypeahead?q=search' to find the best matching geo URNs.

May 3, 2021

Deprecating Connections-Only feature

When creating ugcPosts you have the option of identifying the `visibility` as `PUBLIC` or `CONNECTIONS`. Going forward, we are removing the `CONNECTIONS` scope. Ensure the `visibility` of your ugcPosts follows the format:

JSON

```
    "visibility": {
        "com.linkedin.ugc.MemberNetworkVisibility": "PUBLIC"
    }
```

Additionally, all requests to register a new live event must use the recipe: `urn:li:digitalmediaRecipe:feedshare-live-video`. Any request to POST assets?action=registerLiveEvent should remove the option of using `feedshare-live-video-private`.

New Scheduled Live Events `name` Parameter

We are now providing an easier way for you to manage scheduled live events. A `name` parameter has been added to the liveVideos schema. This field serves to be the new title of the live event, and will eventually replace the existing live event asset title. Example `liveVideos` request below:

JSON

```
{
  "author": {
```

```
    "member": "urn:li:person:iKO_ntS9UQ"
  },
  "scheduledAt": 1591892529000,
  "announcementImage": {
    "media": "urn:li:digitalmediaAsset:C5F22AQEUvl4IOBjE2g",
    "title": {
      "text": "Image title",
      "inferredLocale": "en_US"
    }
  },
  "name": "Scheduled Live Event title"
}
```

Removing Scheduled Live Events Time Restriction

Removed time constraints from liveVideos creation. There is no longer a minimum 1 hour, or 7 day maximum time constraint for new scheduled live events.

Deprecating legacy Live Events feature

Current: Schedule a livestream (scheduled live) AND go live into an event flow (multi-step copy/paste event URL flow) New: ONLY scheduled live flow will exist.

You may have integrated a feature where you ask your members for the URL of their LinkedIn Event, use this URL to obtain an Event URN, and input this Event URN in a ugcPost container. This flow is marked for deprecation in favor of the scheduled live flow.

Refresh Tokens

An often requested feature has been the ability to use OAuth refresh tokens, preventing the need for broadcasters to authorize their account every 60 days. We are now offering this capability to all Live Events Partners. Reach out to your LinkedIn contact to enable this feature on your developer application.

Feedback

Plugins

Article • 05/08/2023

Plugins are a quick and easy way to drop LinkedIn functionality into your website. Use the tools below to automatically generate JavaScript snippets to copy and paste directly to your sites.

- Share
- Follow Company
- LinkedIn AutoFill

By obtaining and using LinkedIn Plugins you agree to the Plugins Terms of Use.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Self-Serve v1 to v2 API Migration Frequently Asked Questions

Article • 05/08/2023

Overview

All new applications created on the LinkedIn Developer Platform as of January 14, 2019 can use LinkedIn's v2 APIs. Starting May 1, 2019, LinkedIn will deprecate use of its v1 APIs. If your developer application currently depends on LinkedIn v1 APIs, see the frequently asked questions below before migrating to LinkedIn v2 APIs.

Does my developer application have access to the LinkedIn v2 API?

All developer applications created on the LinkedIn Developer Portal after January 14, 2019 have access to the LinkedIn v2 API by default. Alternatively, if your developer application has made a successful LinkedIn v1 API request from September 1, 2018 to December 17, 2018, your developer application has immediate access to the v2 API.

What permissions do I have access to?

LinkedIn v1 APIs provided the following set of permissions:

- r_basicprofile
- r_emailaddress
- w_share
- rw_company_admin

Moving forward, the available v2 APIs include:

- r_liteprofile (replaces r_basicprofile)
- r_emailaddress
- w_member_social (replaces w_share)

Looking to maintain access to rw_company_admin? Apply to the LinkedIn Marketing API Program to continue managing your Company Pages.

What are the main differences with the new Sign In with LinkedIn?

With Sign In with LinkedIn, developer applications have access to a member's Basic Profile. In the interest of providing members with greater control over their data, Sign In with LinkedIn using OpenID Connect returns only the critical pieces of member data necessary for identification. The v2 API permits usage of the Lite Profile consisting of a member's id, first name, last name, and profile picture.

A request to retrieve the member profile in v1 may resemble

```
HTTP

GET https://api.linkedin.com/v1/people/~
```

The equivalent request to retrieve the member profile in v2 follows:

```
HTTP

GET https://api.linkedin.com/v2/me
```

Looking to maintain access to the Basic Profile fields? Learn more about LinkedIn Developer Enterprise products.

How do I retrieve the member's email address?

There are no changes to the `r_emailaddress` permission scope used to retrieve the authenticated member's email address. However, the method used to retrieve the email address has been updated.

Whereas the v1 Profile API returned the email address within the Profile response body, a separate request is to retrieve the email address is now required. Use the request below to retrieve the currently authenticated member's email address:

```
HTTP

GET https://api.linkedin.com/v2/emailAddress?q=members&projection=(elements*
(handle~))
```

Troubleshooting

If you have not yet adjusted your application to work around these changes, you will begin to see critical errors occurring now, or in the near future when your authentication tokens next expire. Here are some tips for resolving common potential issues:

1. The most common problem that will occur as a result of the API changes is that your authentication workflow will fail because your app is attempting to request member permissions that you no longer have access to.

   To correct this issue, ensure that the scope parameter in your authorization workflow is no longer requesting any of the following member permissions:

   `r_basicprofile`, `w_share`, `rw_company_admin`.

   Additionally, *default scopes are no longer permitted*. For all OAuth authorization code requests, make sure to include the proper `scope` with your request. The `scope` parameter along with the requested scope must be present within your OAuth request.

   Pay special attention to any 3rd party libraries that you are using for authenticating with LinkedIn, as they may be asking for more member permissions than you realize!

   Note that by removing member permissions, you may also be required to remove API calls that depend on those permissions being present, so you will need to thoroughly review your application and ensure that all of the API calls that it makes can be done under the remaining member permissions.

2. Over the default access token life cycle, even after your LinkedIn API access changes, your application may still have some users with an access token that allows them to call APIs that are no longer available to you.

   When your API access changes, your application may not immediately experience the impact of the changes until your current user access tokens start to expire and you are forced to refresh them to continue making API calls. Please ensure your application is prepared to handle access tokens that were granted before your API access was changed so that they do not request unavailable permissions upon refresh.

Feedback

Profile API

Article • 05/08/2023

> **Note**
>
> The use of this API is restricted to those developers approved by LinkedIn and subject to applicable data restrictions in their agreements.

The Profile API returns a member's LinkedIn profile, subject to the member's privacy settings.

Usage

You must use an access token to make an authenticated call on behalf of a user.

> **Note**
>
> You may only store data returned from the Profile API for the authenticated members with their permission. Please refer to this **document** for guidance on storing authenticated member data. You may never store data returned from the Profile API for members other than the authenticated member.

Retrieve Current Member's Profile

Permissions

This API requires one of the following permissions:

Permission	Description
r_liteprofile	Required to retrieve name and photo for the authenticated user. Please review Lite Profile Fields.
r_basicprofile	Required to retrieve name, photo, headline, and vanity name for the authenticated user. Please review Basic Profile Fields. Note that the v2 r_basicprofile permission grants only a subset of fields provided in v1.
r_compliance	[Private permission] Required to retrieve your activity for compliance monitoring and archiving. This is a private permission and access is granted to select developers.

Request

To identify and retrieve the current member's profile based on the access token, simply call:

https

```
GET https://api.linkedin.com/v2/me
```

Sample Response

JSON


```
{
   "firstName":{
      "localized":{
         "en_US":"Bob"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "localizedFirstName": "Bob",
   "headline":{
      "localized":{
         "en_US":"API Enthusiast at LinkedIn"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "localizedHeadline": "API Enthusiast at LinkedIn",
   "vanityName": "bsmith",
   "id":"yrZCpj2Z12",
   "lastName":{
      "localized":{
         "en_US":"Smith"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "localizedLastName": "Smith",
   "profilePicture": {
        "displayImage": "urn:li:digitalmediaAsset:C4D00AAAAbBCDEFGhiJ"
   }
}
```

Retrieve Other Member's Profile

To retrieve another member's profile, you will need access to the `Person ID`, available only via certain limited access APIs and subject to member privacy settings.

```
https

GET https://api.linkedin.com/v2/people/(id:{person ID})
```

You can also retrieve multiple profiles at once:

```
https

GET https://api.linkedin.com/v2/people?ids=List((id:{Person ID1}),(id:
{Person ID2}),(id:{Person ID3}))
```

> **Note**
>
> In order to make the sample calls above succeed, you must include `X-RestLi-Protocol-Version:2.0.0` in your request header.

This API will only return data for members who haven't limited their Off-LinkedIn Visibility.

Field Selections

By default, only the Lite Profile Fields are returned for a profile request. See the Profile Fields document for a full list of supported fields.

To request more or less fields, you must have additional permissions that are only granted to select partners. Please refer to the field projections on proper syntax. Below is a sample request:

```
https

GET https://api.linkedin.com/v2/people/(id:{profile ID})?projection=
(id,firstName,lastName)
```

Person ID

The `id` returned in the response is the unique identifier of the user. This should be stored and referenced where possible as LinkedIn APIs utilize both URNs and IDs. In our API documentation, we reference this `id` as `person ID`.

> ⓘ Note
>
> Each member `id` is unique to the context of your application only. Sharing a `person ID` across applications will not work and result in a 404 error.

Public Profile URL

The `vanityName` from Basic Profile Fields is used to represent the public profile URL in the follow format: `www.linkedin.com/in/{vanityName}`.

New Location Display Name

The `geoLocation` from Location Fields is the new location field. As LinkedIn transitions through the Bing Geo location migration, we will try to maintain backwards compatibility with the legacy location and `locationName` field as much as possible.

> ⓘ Note
>
> All use of the Microsoft Bing Maps location data is subject to **Microsoft Bing Maps and MapPoint Web Service End User Terms of Use** and the **Microsoft Privacy Statement**. By accessing any Microsoft Bing Maps location data, you are agreeing to be bound by these Microsoft terms.

To determine a member's profile location, refer to the geoLocation field. If the `autoGenerated` field is `false`, then the member's location has already migrated to Bing Geo taxonomy. This means that the most up-to-date display name is retrieved from `geo` field in `geoLocation`. If the field is `true`, then you can rely on either `location` or `geo` in `geoLocation`.

In order to get the display name from the geo URN value of `geo` field, please use the Geo API. Alternatively, you can utilize decoration in your Profile request:

```
https
```

```
GET https://api.linkedin.com/v2/me?projection=
(geoLocation(geo~,autoGenerated))

GET https://api.linkedin.com/v2/people/(id:{person ID})?projection=
(geoLocation(geo~,autoGenerated))
```

JSON


```
{
  "geoLocation": {
    "geo": "urn:li:geo:12345",
    "autoGenerated": false,
    "geo~": {
      "id": 12345,
      "defaultLocalizedName": {
        "locale": {
          "country": "US",
          "language": "en"
        },
        "value": "San Francisco Bay Area"
      }
    }
  }
}
```


Legacy Location Display Name

The `location` from Profile Fields contains several fields that are used to determine the member's displayed location nam

If the `userSelectedGeoPlaceCode` is present, then you will need to call Places API - GET to retrieve the name. To use the API, you will need to translate the `countryCode` to a countryURN by simply appending `urn:li:country:` in front of the code. See below for an example:

JSON


```
{
   "location":{
      "postalCode":"12345",
      "standardizedLocationUrn":"urn:li:standardizedLocationKey:(us,12345)",
      "userSelectedGeoPlaceCode":"1-1-0-23-30",
      "countryCode":"us"
   }
}
```

https

```
GET
https://api.linkedin.com/v2/places/country=urn:li:country:us&placeCode=1-1-
0-23-30
```

If the `userSelectedGeoPlaceCode` is **NOT** present, then you will need to call Regions API - FINDER standardizedLocation to retrieve the name. To use the API, you will input the `standardizedLocationUrn` value into the `standardizedLocation` parameter. See below for an example:

JSON

```
{
   "location":{
      "postalCode":"12345",
      "standardizedLocationUrn":"urn:li:standardizedLocationKey:(us,12345)",
      "countryCode":"us"
   }
}
```

https

```
GET https://api.linkedin.com/v2/regions?
q=standardizedLocation&standardizedLocation=urn:li:standardizedLocationKey:
(us,12345)
```

Once you make the appropriate request, you can simply retrieve the display location name from the `value` of the `name` field for each respective API.

Feedback

Lite Profile Fields

Article • 05/08/2023

To access any of the Lite Profile fields listed below, your application must request the `r_liteprofile` member permission scope:

Field Name	Description
id	The unique identifier for the given member. May also be referenced as the `personId` within a Person URN (urn:li:person:{personId}). The `id` is unique to your specific developer application. Any attempts to use the `id` with other developer applications will not succeed.
firstName	First name of the member. Represented as a MultiLocaleString object type.
localizedFirstName	Localized version of the firstName field.
lastName	Last name of the member. Represented as a MultiLocaleString object type.
localizedLastName	Localized version of the lastName field.
profilePicture	Metadata about the member's picture in the profile. See Profile Picture Fields for more information.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Basic Profile Fields

Article • 05/08/2023

Field Name	Description
id	A unique identifying value for the member. Referenced as personId in other API documentation pages. Can also be referenced as an URN such as urn:li:person:{personId}. This value is linked to your specific application. Any attempts to use it with a different application will result in a 404 error.
firstName	Localizable first name of the member. Represented as a MultiLocaleString object type.
localizedFirstName	Localized version of the firstName field.
lastName	Localizable last name of the member. Represented as a MultiLocaleString object type.
localizedLastName	Localized version of the lastName field.
maidenName	Localizable maiden name of the member. Represented as a MultiLocaleString object type.
localizedMaidenName	Localized version of the maidenName field.
headline	Localizable headline of member's choosing. Represented as a MultiLocaleString object type.
localizedHeadline	Localized version of the headline field.
profilePicture	Metadata about the member's picture in the profile. This will replace pictureInfo. See Profile Picture Fields for more information.
vanityName	The vanity name of the member. Vanity name is represented as a string is used for the public profile URL: `www.linkedin.com/in/{vanityName}`.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Profile Picture Fields

Article • 05/08/2023

Field Name	Description
created	An epoch timestamp corresponding to the creation of the picture. Represented in milliseconds. This field requires special permissions available only to select partners.
deleted	An epoch timestamp corresponding to the deletion of the picture. Represented in milliseconds. Optional. This field requires special permissions available only to select partners.
displayImage	The digitalmediaAsset URN of the display image.
lastModified	An epoch timestamp corresponding to the last modification of the picture. Represented in milliseconds. This field requires special permissions available only to select partners.
originalImage	The digitalmediaAsset URN of the original image. Will only appear in self-view. This field requires special permissions available only to select partners.
photoFilterEditInfo	Optional information object on the photo filter operations applied to the profile picture. This field requires special permissions available only to select partners.

To decorate the `digitalMediaAsset` URN from `displayImage` or `originalImage` , you will use parameter `projection` that will include `~digitalmediaAsset:playableStreams` . For more information, see here about digital media asset URNs . See below for an example of Profile API:

https

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

profile api sample response

JSON

```
{
   "profilePicture":{
      "displayImage":"urn:li:digitalmediaAsset:C4D03AQGsitRwG8U8ZQ",
      "displayImage~":{
         "elements":[
```

```
            {
               "artifact":"urn:li:digitalmediaMediaArtifact:
(urn:li:digitalmediaAsset:C4D03AQGsitRwG8U8ZQ,urn:li:digitalmediaMediaArtifa
ctClass:profile-displayphoto-shrink_100_100)",
               "authorizationMethod":"PUBLIC",
               "data":{
                  "com.linkedin.digitalmedia.mediaartifact.StillImage":{
                     "storageSize":{
                        "width":100,
                        "height":100
                     },
                     "storageAspectRatio":{
                        "widthAspect":1,
                        "heightAspect":1,
                        "formatted":"1.00:1.00"
                     },
                     "mediaType":"image/jpeg",
                     "rawCodecSpec":{
                        "name":"jpeg",
                        "type":"image"
                     },
                     "displaySize":{
                        "uom":"PX",
                        "width":100,
                        "height":100
                     },
                     "displayAspectRatio":{
                        "widthAspect":1,
                        "heightAspect":1,
                        "formatted":"1.00:1.00"
                     }
                  }
               },
               "identifiers":[
                  {

"identifier":"https://media.licdn.com/dms/image/C4D03AQGsitRwG8U8ZQ/profile-
displayphoto-shrink_100_100/0?e=1526940000&v=alpha&t=12345",
                     "file":"urn:li:digitalmediaFile:
(urn:li:digitalmediaAsset:C4D03AQGsitRwG8U8ZQ,urn:li:digitalmediaMediaArtifa
ctClass:profile-displayphoto-shrink_100_100,0)",
                     "index":0,
                     "mediaType":"image/jpeg",
                     "identifierExpiresInSeconds":1526940000
                  }
               ]
            }
         ],
         "paging":{
            "count":10,
            "start":0,
            "links":[

            ]
         }
```

```
        }
    },
    "id":"yrZCpj2ZYQ"
}
```

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Location Field

Article • 05/08/2023

geoLocation Field (New)

Member's current location using Bing Geo taxonomy. This field is currently optional while Bing Geo migration is in progress.

Field Name	Description	Type
geo	Member's current location defined by Bing Geo URN.	Geo URN
autoGenerated	Default to true. True if existing `geoLocation` is back filled or dual written from legacy `location`. False if it's a member input. This is read-only, temporary and used for Bing Geo migration to indicate whether the existing geo location is a member input. Clients may suggest members to re-select their location when they haven't.	boolean

location Field (Legacy)

Field Name	Type
countryCode	The lower-case ISO 3166-1 alpha-2 standard country code representing the member's current country.
postalCode	Optional string representing the postal code of the location.
standardizedLocationUrn	An optional standardized referenced location URN.
userSelectedGeoPlaceCode	Optional string representing the place code of the geographic location.

Note: To understand how the display name of the member's location is determined, please refer here.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Position Field

Article • 05/08/2023

Field Name	Sub-Field Name	Required	Description
id		Yes	The unique identifier of the position object.
company		No	Standardized referenced company URN.
companyName		No	Localizable company name as entered by the member. It is a MultiLocaleString type.
description		No	Localizable description for this position. It is a [MultiLocaleRichText](../object-types.md#multilocalerichtext type.
endMonthYear		No	Last month and year at this position. Missing value means the position is current. It is a [Date](../object-types.md#date type. Does not support "day" field.
location		No	Legacy location for the position. Only displayed if locationName field is empty.
	countryCode	Yes	2 letter country code. Refer to the standardized country URNs for more information.
	regionCode	No	Optional integer code. Refer to the standardized region URNs for more information.
locationName		No	Legacy localizable location name of the position. It is a MultiLocaleString type.
memberRichContents		No	The list of MemberRichContentUrn associated with the education. Default to empty array.
startMonthYear		No	Start month and year at this position. It is a Date type. Does not support "day" field.

Field Name	Sub-Field Name	Required	Description
title		No	Localizable title name of the position. It is a MultiLocaleString type.
geoPositionLocation		No	New location of the position member worked or works at. This field is absent if member doesn't input position location.
	displayLocationName	Yes	Location of the position as selected from typeahead or entered by the member. This field is a MultiLocaleString type. Validations enforced are: 1) the keys in a localized map all exist within the profile's supportedLocale set; 2) there is a value for profile default locale in the localized string maps.

Feedback

Background Picture Fields

Article • 05/08/2023

Field Name	Description
created	An epoch timestamp corresponding to the creation of the picture. Represented in milliseconds.
deleted	An epoch timestamp corresponding to the deletion of the picture. Represented in milliseconds. Optional.
displayImage	The digitalmediaAsset URN of the display image.
lastModified	An epoch timestamp corresponding to the last modification of the picture. Represented in milliseconds.
originalImage	The digitalmediaAsset URN of the original image. Will only appear in self-view.
photoFilterEditInfo	Optional information object on the photo filter operations applied to the background picture.

To decorate the `digitalMediaAsset` URN from `displayImage` or `originalImage` , you will use parameter `projection` that will include `~:playableStreams` . For more information, see here about digital media asset URNs . See below for an example of Profile API:

https

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

profile api sample responses

JSON

```
{
   "backgroundPicture":{
      "displayImage":"urn:li:digitalmediaAsset:C4D03AQGsitRwG8U8ZQ",
      "displayImage~":{
         "elements":[
            {
               "artifact":"urn:li:digitalmediaMediaArtifact:
(urn:li:digitalmediaAsset:C4D03AQGsitRwG8U8ZQ,urn:li:digitalmediaMediaArtifa
ctClass:profile-displayphoto-shrink_100_100)",
               "authorizationMethod":"PUBLIC",
               "data":{
```

```
                  "com.linkedin.digitalmedia.mediaartifact.StillImage":{
                     "storageSize":{
                        "width":100,
                        "height":100
                     },
                     "storageAspectRatio":{
                        "widthAspect":1,
                        "heightAspect":1,
                        "formatted":"1.00:1.00"
                     },
                     "mediaType":"image/jpeg",
                     "rawCodecSpec":{
                        "name":"jpeg",
                        "type":"image"
                     },
                     "displaySize":{
                        "uom":"PX",
                        "width":100,
                        "height":100
                     },
                     "displayAspectRatio":{
                        "widthAspect":1,
                        "heightAspect":1,
                        "formatted":"1.00:1.00"
                     }
                  }
               },
               "identifiers":[
                  {

"identifier":"https://media.licdn.com/dms/image/C4D03AQGsitRwG8U8ZQ/profile-displayphoto-shrink_100_100/0?e=1526940000&v=alpha&t=12345",
                     "file":"urn:li:digitalmediaFile:(urn:li:digitalmediaAsset:C4D03AQGsitRwG8U8ZQ,urn:li:digitalmediaMediaArtifactClass:profile-displayphoto-shrink_100_100,0)",
                     "index":0,
                     "mediaType":"image/jpeg",
                     "identifierExpiresInSeconds":1526940000
                  }
               ]
            }
         ],
         "paging":{
            "count":10,
            "start":0,
            "links":[

            ]
         }
      }
   },
   "id":"yrZCpj2ZYQ"
}
```

> **Note**
>
> Retrieving media via the LinkedIn CDN requires converting URNs to URLs as described in the examples below using stock photos provided by LinkedIn. If the media-typed URN starts with `urn:li:media:`, you will need to replace that with `https://media.licdn.com/mpr/mpr` . For example, `urn:li:media:/gcrc/dms/image/ABC/background-displayphoto-shrink_800_800/0?e=1525021200&v=alpha&t=abc123` would translate to `https://media.licdn.com/mpr/mpr/gcrc/dms/image/ABC/background-displayphoto-shrink_800_800/0?e=1525021200&v=alpha&t=abc123`. If the media-typed URN starts with `urn:li:scds:`, you will need to replace that with `https://static.licdn.com/scds/common/u/`. For example, `urn:li:scds:images/apps/profile/topcard_backgrounds/texture_bokeh_1400x425.jpg` would translate to `https://static.licdn.com/scds/common/u/images/apps/profile/topcard_backgrounds/texture_bokeh_1400x425.jpg`.

Feedback

Certification Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The unique identifer of the certification object.
authority	No	Localizable name of the certification's issuing body. It is a MultiLocaleString type.
company	No	Standardized referenced company URN.
endMonthYear	No	End date for the certification. It is a Date type. Does not support "day" field.
licenseNumber	No	Localizable license number for the certification. It is a MultiLocaleString type.
name	No	Localizable certification name. It is a MultiLocaleString type.
startMonthYear	No	Start date for the certification. It is a Date type. Does not support "day" field.
url	No	External reference to the certification's website or program.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Course Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The unique identifier of the course object.
name	No	Localizable certification name. It is a MultiLocaleString type.
number	No	Assigned course number. Represented in string.
occupation	No	Member's occupation when the course was completed. Represented as either a standardized referenced company or school URN.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Profile Education Fields

Article • 05/08/2023

Field Name	Sub-Field Name	Required	Description
id		Yes	The unique identifier of the education object.
activities		No	Localizable description of activities, societies, clubs, or sports teams the member participated in while attending this school. It is a MultiLocaleString type.
degreeName		No	Localizable degree attained at this school. It is a MultiLocaleString type.
endMonthYear		No	End date of the education. It is a Date type. Does not support "day" and "month" field.
fieldsOfStudy		No	Degrees achieved in respective fields of study.
	fieldOfStudyName	No	Localizable field of study or major. It is a MultiLocaleString type.
grade		No	Grade attained in the area of study.
	grade	No	Localizable grade of the degree or major. It is a MultiLocaleString type.
memberRichContents		No	The list of MemberRichContentUrn associated with the education. Default to empty array.
notes		No	Localizable description for additional details about this education. It is a MultiLocaleRichText type.
organization		No	Organization URN used to represent a school.
program		No	Standardized referenced school program URN.
school		No	*Deprecated*. Please use `organization`. Standardized referenced School URN.

Field Name	Sub-Field Name	Required	Description
schoolName		No	Localizable school name. It is a MultiLocaleString type.
startMonthYear		No	Start date of the education. It is a Date type. Does not support "day" and "month" field.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Honor Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The unique identifier of the honor object.
description	No	Localizable description of the honor or award. It is a MultiLocaleRichText type.
issueDate	No	Month and year the honor or award was issued. It is a Date type. Does not support "day" field.
issuer	No	Localizable entity that issued the honor or award. It is a MultiLocaleString type.
occupation	No	Member's occupation when the honor or award was completed. Referenced as either a standardized referenced company or school URN.
title	Yes	Localizable title of the honor or award. It is a MultiLocaleString type.

Feedback

IM Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The IM name represented in string.
provider	Yes	Enum of the IM provider: • AIM (AIM) - `Deprecated` • SKYPE (Skype) • YAHOO (Yahoo Messenger) - `Deprecated` • ICQ (ICQ) • GTALK (Google Talk) • QQ (QQ) • WEIXIN (WeChat)

Feedback

Language Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The unique identifier of the language object.
name	Yes	Localizable language name. It is a MultiLocaleString type.
proficiency	No	Member's proficiency level of the language. Possible enum values: • ELEMENTARY (Elementary proficiency) • LIMITED_WORKING (Limited working proficiency) • PROFESSIONAL_WORKING (Professional working proficiency) • FULL_PROFESSIONAL (Full professional proficiency) • NATIVE_OR_BILINGUAL (Native or bilingual proficiency)

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Organization Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The unique identifier of the organization object.
description	No	Localizable description for the position held at the organization. It is a MultiLocaleRichText type.
endMonthYear	No	Month and year start date at the organization. It is a Date type. Does not support "day" field.
name	No	Localizable name of the organization. It is a MultiLocaleString type.
occupation	No	Member's occupation when the course was completed. Referenced as either a standardized referenced company or school URN.
position	No	Localizable name of the position at the organization. It is a MultiLocaleString type.
startMonthYear	No	Start date for the certification. It is a Date type. Does not support "day" field.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Patent Fields

Article • 05/08/2023

Field Name	Sub-Field Name	Required	Description
id		Yes	The unique identifier of the patent object.
applicationNumber		No	Localizable patent application number. It is a MultiLocaleString type. Only displayed when pending is true.
description		No	Localizable description for additional details about this education. It is a MultiLocaleRichText type.
filingDate		No	Month, day, and year the patent was filed. It is a Date type. Only displayed when pending is true.
inventors		Yes	Members who created the patent or contributed to it. Represented in an array of Contributors. Required to have the member's own person URN in the array.
	memberId	No	The inventor represented in person URN.
	name	No	Localizable member name. It is a MultiLocaleString type.
issueDate		No	Month, day, and year the patent was officially issued. It is a Date type. Only displayed when pending is false.
issuer		Yes	Localizable issuer of the patent. Issuer based on country and is represented by lowercase two-letter country code as defined by ISO-3166 . It is a MultiLocaleString type.
number		No	Localizable patent number. It is a MultiLocaleString type. Only displayed when pending is false.
pending		Yes	The status of patent represented as a boolean.
title		Yes	Localizable title of the patent. It is a MultiLocaleString type.
url		No	URL referencing the patent represented in string.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Phone Number Fields

Article • 05/08/2023

These phone numbers are listed under the "Contact Info" of the profile section.

Field Name	Description
number	Formatted String of the phone number.
type	Optional enum: • WORK • HOME • MOBILE • FAX • PAGER

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Project Fields

Article • 05/08/2023

Field Name	Sub-Field Name	Required	Description
id		Yes	The unique identifier of the publication object.
description		No	Localizable description of the project. It is a MultiLocaleRichText type.
endMonthYear		No	Month and year indicating when the project ended. It is a Date type. Does not support "day" field.
members		Yes	People who contributed to the project. Represented in an array of Contributors. Required to have the member's own person URN in the array.
	memberId	No	The contributor represented in person URN.
	name	No	Localizable member name. It is a MultiLocaleString type.
occupation		No	Position a member held while working on this project. Selected from a position of the member's profile. Represented as either a standardized referenced company or school URN.
singleDate		No	A boolean that distinguishes between an ongoing project without an end date and a project that occurred at one specofic time.
startMonthYear		No	Start date for the certification. It is a Date type. Does not support "day" field.
title		Yes	Localizable name of the project. It is a MultiLocaleString type.
url		No	URL referencing the project represented in string.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Publication Fields

Article • 05/08/2023

Field Name	Sub-Field Name	Required	Description
id		Yes	The unique identifier of the publication object.
authors		Yes	People who wrote the publication or contributed to it. Represented in an array of Contributors. Required to have the member's own person URN in the array.
	memberId	No	The contributor represented in person URN.
	name	No	Localizable member name. It is a MultiLocaleString type.
date		No	Day, month, and year indicating when the publication was published. It is a Date type.
description		No	Localizable description of the publication. It is a MultiLocaleRichText type.
name		Yes	Localizable name of the publication. It is a MultiLocaleString type.
publisher		No	Localizable publication or publisher for the title. It is a MultiLocaleString type.
url		No	URL referencing the publication represented in string.

Skill Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The unique identifier of the skill object.
name	Yes	Localizable skill name as defined by the member. It is a MultiLocaleString type.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Test Score Fields

Article • 05/08/2023

Field Name	Required	Description
date	No	Month and year the test was taken. It is a Date type. Does not support "day" field.
description	No	Localizable description of the test score. It is a MultiLocaleRichText type.
name	Yes	Localizable name of the test. It is a MultiLocaleString type.
occupation	No	Position a member held while during this test. Selected from a position of the member's profile. Represented as either a standardized referenced company or school URN.
score	No	Score achieved on the test represented in string.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Volunteering Experience Fields

Article • 05/08/2023

Field Name	Required	Description
id	Yes	The unique identifier of the volunteering experience object.
cause	No	Cause of the volunteering experience represented in predefined string. Enum of predefined volunteering causes: • animalRights • artsAndCulture • children • civilRights • economicEmpowerment • education • environment • health • humanRights • humanitarianRelief • politics • povertyAlleviation • scienceAndTechnology • socialServices
company	No	Standardized referenced company URN.
companyName	Yes	Localizable company name. It is a MultiLocaleString type.
description	No	Localizable description of the experience. It is a MultiLocaleRichText type.
endMonthYear	No	Month and year end date of the experience. It is a Date type. Does not support "day" field.
role	Yes	Localizable duty or responsibility performed at this volunteering experience. It is a MultiLocaleString type.
singleDate	No	A boolean that distinguishes between an ongoing volunteering experience without an end date and a volunteering experience that occurred at one specofic time.
startMonthYear	No	Month and year start date of the experience. It is a Date type. Does not support "day" field.

Feedback

Volunteering Interest Fields

Article • 05/08/2023

Field Name	Sub-Field Name	Required	Description
supportedNonprofits		No	DEPRECATED. Use "selectedContactInterests" instead. Optional array of SupportedNonprofit.
	companyId	No	A standardized referenced company URN. Refer here for more information.
	companyName	Yes	Localizable company name. It is a MultiLocaleString object type.
supportedPredefinedCauses		No	An array of enum. Enum of predefined volunteering causes: • animalRights • artsAndCulture • children • civilRights • economicEmpowerment • education • environment • health • humanRights • humanitarianRelief • politics • povertyAlleviation • scienceAndTechnology • socialServices
supportedUserDefinedCauses		No	An array of user inputted string. Not currently used in any LinkedIn platform's UI.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Website Fields

Article • 05/08/2023

Field Name	Sub-Field Name	Required	Description
category		No	Enum of predefined volunteering causes: • PERSONAL • COMPANY • BLOG • RSS • PORTFOLIO • OTHER
label		No	Localizable label a member chose for a website in "OTHER" category. It is a MultiLocaleString type.
url		Yes	Localized URLs for a website. It is a MultiLocaleUrl type.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Picture Info Fields

Article • 05/08/2023

This field will be deprecated on December 1, 2018. Please refer to Profile Picture fields and Media Migration Guide for more information.

Field Name	Sub-Field Name	Description
cropInfo		The metadata of the cropped image.
	height	Height represented in int.
	width	Width represented in int.
	x	X coordinate of upper-left corner of the cropped area (upper-left corner of image is 0). Represented in int.
	y	Y coordinate of upper-left corner of the cropped area (upper-left corner of image is 0). Represented in int.
croppedImage		Profile image cropped for display.
hidden		Optional boolean. Defaults to false.
masterImage		The original raw image uploaded by the member. This will not be used for display and will only be used for re-cropping purposes. Only available for self lookup.
state		Optional state defaulting to "ACTIVE". Can be the following values: • ACTIVE • DELETED • ABUSIVE • QUESTIONED

Note: To get the media, replace the media-typed URN sub-string `urn:li:media:` to `https://media.licdn.com/mpr/mpr` . For example, if the field value is `urn:li:media:/gcrc/dms/image/ABC/profile-displayphoto-shrink_800_800/0?e=1525021200&v=alpha&t=abc123` , you can modify it to `https://media.licdn.com/mpr/mpr/gcrc/dms/image/ABC/background-profile-shrink_800_800/0?e=1525021200&v=alpha&t=abc123` to retrieve it as a media URL via the LinkedIn CDN.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Background Image Fields

Article • 05/08/2023

This field wae deprecated on December 1, 2018. Please refer to Background Picture fields and Media Migration Guide for more information.

Field Name	Sub-Field Name	Type
cropInfo		Stores the top-Y for member uploaded, repositioned images.
	height	int
	width	int
	x	X coordinate of upper-left corner of the cropped area (upper-left corner of image is 0). Represented in int.
	y	Y coordinate of upper-left corner of the cropped area (upper-left corner of image is 0). Represented in int.
pictureId		The media URN of the background image. Refer to media gateway for more information.
croppedPictureId		The media URN of the cropped background image. Refer to media gateway for more information.
customUpload		A required representing whether the image was a member upload. Always true.

Note

Retrieving media via the LinkedIn CDN requires converting URNs to URLs as described in the examples below using stock photos provided by LinkedIn.

- If the media-typed URN starts with `urn:li:media:`, you will need to replace that with `https://media.licdn.com/mpr/mpr` . For example, `urn:li:media:/gcrc/dms/image/ABC/background-displayphoto-shrink_800_800/0?e=1525021200&v=alpha&t=abc123` would translate to `https://media.licdn.com/mpr/mpr/gcrc/dms/image/ABC/background-displayphoto-shrink_800_800/0?e=1525021200&v=alpha&t=abc123`.
- If the media-typed URN starts with `urn:li:scds:`, you will need to replace that with `https://static.licdn.com/scds/common/u/`. For example,

`urn:li:scds:images/apps/profile/topcard_backgrounds/texture_bokeh_1400x425.jpg` would translate to `https://static.licdn.com/scds/common/u/images/apps/profile/topcard_backgrounds/texture_bokeh_1400x425.jpg`.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Object Types

Article • 05/08/2023

MultiLocaleString

The `MultiLocaleString` object represents a textual fields with values for multiple locales. It contains two fields: `localized` and `preferredLocale` . See below for a detailed description of these fields, allowed values, and sample JSON.

MultiLocaleString Schema

Field Name	Sub-Field Name	Required	Description
localized		Yes	Maps a locale to a localized version of the string. Each key is a Locale record converted to string format, with the language, country and variant separated by underscores. Examples: 'en', 'de_DE', 'en_US_WIN', 'de_POSIX', 'fr_MAC'.
preferredLocale		No	The preferred locale to use, based on standard rules.
	country	No	An uppercase two-letter country code as defined by ISO-3166 .
	language	Yes	A lowercase two-letter language code as defined by ISO-639 .

Sample JSON of multiLocaleString

JSON

```
{
   "address":{
      "localized":{
         "en_US":"2029 Stierlin Ct, Mountain View, CA 94043"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   }
}
```

MultiLocaleRichText

The `MultiLocaleRichText` object represents a textual fields with values for multiple locales. It contains two fields: `localized` and `preferredLocale` . See below for a detailed description of these fields, allowed values, and sample JSON.

MultiLocaleRichText Schema

Field Name	Sub-Field Name	Required	Description
localized		Yes	Maps a locale to a localized version of the string. Each key is a Locale record converted to string format, with the language, country and variant separated by underscores. Examples: 'en', 'de_DE', 'en_US_WIN', 'de_POSIX', 'fr_MAC'.
	rawText	No	Rich text represented in string.
preferredLocale		No	The preferred locale to use, based on standard rules.
	country	No	An uppercase two-letter country code as defined by ISO-3166 .
	language	Yes	A lowercase two-letter language code as defined by ISO-639 .

Sample JSON of multiLocaleRichText

JSON

```
{
   "summary":{
      "preferredLocale":{
         "country":"US",
         "language":"en"
      },
      "localized":{
         "en_US":{
            "rawText":"Awesome summary of me."
         }
      }
   }
}
```

MultiLocaleUrl

The `MultiLocaleUrl` object represents a textual fields with values for multiple locales. It contains two fields: `localized` and `preferredLocale` . See below for a detailed description of these fields, allowed values, and sample JSON.

MultiLocaleUrl Schema

Field Name	Sub-Field Name	Required	Description
localized		Yes	Maps a locale to a localized version of the string. Each key is a Locale record converted to string format, with the language, country and variant separated by underscores. Examples: 'en', 'de_DE', 'en_US_WIN', 'de_POSIX', 'fr_MAC'.
preferredLocale		No	The preferred locale to use, based on standard rules.
	country	No	An uppercase two-letter country code as defined by ISO-3166 .
	language	Yes	A lowercase two-letter language code as defined by ISO-639 .

Sample JSON of multiLocaleUrl

JSON

```
{
   "url":{
      "localized":{
         "en_US":"http://www.linkedin-other.com"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   }
}
```

Date

The `Date` object represents a date of day, month and year.

Date Schema

Field Name	Required	Type
day	No	Day represented in integer. Valid range from 1 to 31 depending on month.
month	No	Month represented in integer. Valid range from 1 to 12.
year	No	Year represented in integer.

Sample JSON of date

JSON

```
{
   "birthDate":{
      "day":1,
      "month":1,
      "year":1974
   }
}
```

LocaleString

The `LocaleString` object represents a textual fields with the specified locale. It contains two fields: `locale` and `value` . See below for a detailed description of these fields, allowed values, and sample JSON.

LocaleString Schema

Field Name	Sub-Field Name	Required	Description
locale		Yes	Maps a locale to a localized version of the string. Each key is a Locale record converted to string format, with the language, country and variant separated by underscores. Examples: 'en', 'de_DE', 'en_US_WIN', 'de_POSIX', 'fr_MAC'.
	country	No	An uppercase two-letter country code as defined by ISO-3166 .
	language	Yes	A lowercase two-letter language code as defined by ISO-639 .
value		Yes	The value represented in string.

Sample JSON of LocaleString

JSON

```
{
   "name":{
      "locale":{
         "country":"US",
         "language":"en"
      },
      "value":"California"
   }
}
```

Locale

The `Locale` object represents a `country` and `language` . See below for a detailed description of these fields, allowed values, and sample JSON.

Locale Schema

Field Name	Required	Description

Field Name	Required	Description
country	No	An uppercase two-letter country code as defined by ISO-3166 .
language	Yes	A lowercase two-letter language code as defined by ISO-639 .

Sample JSON of Locale

JSON

```
{
   "locale":{
      "country":"US",
      "language":"en"
   }
}
```

Video

The `Video` object represents the metadata for a video content.

Video Schema

Field Name	Required	Format	Description
description	Optional	MultiLocaleString	Description of this video.
duration	Optional	int	Duration of the video in milliseconds.
height	Yes	int	Height of this video in pixels.
htmlEmbedCode	Optional	String	The html embed code for this video.
title	Optional	MultiLocaleString	Title of this video.
url	Yes	String	The external url of this video.
width	Yes	int	Width of this video in pixels.

Sample JSON of Video

JSON

```
{
   "description":{
```

```
      "localized":{
         "en_US":"Description of the video"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "duration":60000,
   "height":720,
   "title":{
      "localized":{
         "en_US":"Title of the video"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "url":"https://linkedin.com",
   "width":1080
}
```


Image

The `Image` object represents the metadata for an image content.

Image Schema

Field Name	Required	Format	Description
description	Optional	MultiLocaleString	Description of this image.
height	Yes	int	Height of this image in pixels.
htmlEmbedCode	Optional	String	The html embed code for this image.
slideShareImage	Optional	SlideShareImage Urn	The urn to slideshow for this image if uploaded to SlideShare.
title	Optional	MultiLocaleString	Title of this image.
url	Yes	String	The external url of this image.
width	Yes	int	Width of this image in pixels.

Sample JSON of Image

JSON


```
{
   "description":{
      "localized":{
         "en_US":"Description of the image"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "height":720,
   "title":{
      "localized":{
         "en_US":"Title of the image"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "url":"https://linkedin.com",
   "width":1080
}
```


Document

The `Document` object represents the metadata for a document content. The document is in text files, e.g., .pdf, .doc, .ppt, .pps, .rtf.

Image Schema

Field Name	Required	Format	Description
description	Optional	MultiLocaleString	Description of this document.
height	Yes	int	Height of this document in pixels.
htmlEmbedCode	Optional	String	The html embed code for this document.
slideshow	Optional	SlideShareSlideShowUrn	The urn to slideshow for this document if uploaded to SlideShare.
title	Optional	MultiLocaleString	Title of this document.
url	Yes	String	The external url of this document.

Field Name	Required	Format	Description
width	Yes	int	Width of this document in pixels.

Sample JSON of Document

JSON

```
{
   "description":{
      "localized":{
         "en_US":"Description of the document"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "height":720,
   "title":{
      "localized":{
         "en_US":"Title of the document"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "url":"https://linkedin.com",
   "width":1080
}
```

Link

The `Link` object represents the metadata for a link content. The link contains a url that doesn not point to an image, video or document.

Link Schema

Field Name	Required	Format	Description
description	Optional	MultiLocaleString	Description of this link.
title	Optional	MultiLocaleString	Title of this link.
url	Yes	String	The external url of this link.

Sample JSON of Link

JSON

```
{
   "description":{
      "localized":{
         "en_US":"Description of the document"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "title":{
      "localized":{
         "en_US":"Title of the document"
      },
      "preferredLocale":{
         "country":"US",
         "language":"en"
      }
   },
   "url":"https://linkedin.com"
}
```

AuditStamp

The `AuditStamp` object represents the metadata of when an object is acted upon.

AuditStamp Schema

Field Name	Required	Format	Description
actor	Yes	Urn	The entity authorized the change.
impersonator	Optional	Urn	The entity which performs the change on behalf of the actor. Must be authorized to act as the actor.
time	Yes	long	When the event happened in epoch time.

Sample JSON of AuditStamp

JSON

```
{
   "actor":"urn:li:person:123ABC",
```

```
    "time":1332187798000
}
```

CroppedImage

An image with its cropping information.

CroppedImage Schema

Field Name	Required	Format	Description
cropped	Yes	Urn	Location of the cropped image.
original	Yes	Urn	Location of the original image.
cropInfo	Yes	Rectangle	The cropping information defined by a rectangle in which the specified corner is the upper-left corner of the crop area.

###Sample JSON of CroppedImage

JSON

```
{
  "cropped": "urn:li:media:/gcrc/dms/image/Z561BAQF8zT9beoHupA/company-
background_10000/0/1544648748713?
e=1631905200&v=beta&t=ZSHOeHDliP1QEmnMoIg3GPUcLocCXMx0OaU_X2MJNks",
  "original": "urn:li:media:/gcrc/dms/image/Z561BAQF8zT9beoHupA/company-
background_10000/0/1544648748713?
e=1631905200&v=beta&t=ZSHOeHDliP1QEmnMoIg3GPUcLocCXMx0OaU_X2MJNks",
  "cropInfo": {
    "x": 0,
    "width": 646,
    "y": 0,
    "height": 220
  }
}
```

Rectangle

An abstract rectangle defined by the coordinates of a corner and its width and height without any associated domain specific semantics.

Rectangle Schema

Field Name	Required	Format	Description
height	Yes	int	Height of the image.
width	Yes	int	Width of the image.
x	Yes	int	X coordinate of the corner.
y	Yes	int	Y coordinate of the corner.

###Sample JSON of Rectangle

JSON

```
{
  "x": 0,
  "width": 646,
  "y": 0,
  "height": 220
}
```

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A