Token Introspection

02/22/2022

Contents

[Introduction 1](#_Toc96422563)

[Prerequisites 2](#_Toc96422564)

[Working with Introspection Endpoint 2](#_Toc96422565)

[Authorization 2](#_Toc96422566)

[Approach 1: Token Introspection for RestliGateway users 2](#_Toc96422567)

[Benefits 4](#_Toc96422568)

[Metadata 4](#_Toc96422569)

[Token Introspection Request 4](#_Toc96422570)

[Sample Request 5](#_Toc96422571)

[Token Introspection Response 5](#_Toc96422572)

[Sample Responses 6](#_Toc96422573)

[References 7](#_Toc96422574)

# Introduction

This document captures the steps to fetch metadata of an access/refresh token.

Audience

Partner engineering team.

Use cases

The procedures captured in this document allow you to:

1. Look up metadata of a token such as permissions, token TTL, and so on
2. Know the scope associated with the token you are using
3. Use the token introspection endpoints

**Note:** Currently, token introspection endpoint is available in the Partnering tool. Exposing a similar endpoint to help partners can resolve any permission related issues.

Benefits

1. Easy permissions migration
2. The Token introspection endpoint is useful to programmatically generate access token using refresh token when access token expires. You can check TTL to go through the refresh token flow and get a new AT, instead of trying to use an expired access token.
3. Decrease in calls or emails related to failing API with permission issues

# Prerequisites

The given access or refresh token for which you want to fetch the metadata. If you do not have one, refer to the procedure (to cross refer the content) to generate the token.

# Working with Introspection Endpoint

The objective is to return metadata information if you have the access/refresh token. This is required both in RestliGateway and GraphQLGateway.

As per IETF RFC recommendation, the protected resource calls the introspection endpoint using an HTTP POST request with parameters sent as "application/x-www-form-urlencoded" data.

## Authorization

To prevent token scanning attacks, RFC recommends the following ways of authorization to access this endpoint:

1. Client authentication as described in OAuth 2.0 [[RFC6749](https://tools.ietf.org/html/rfc6749)]
2. Separate OAuth 2.0 access token such as the bearer token

The approach 1 below is using client authentication and approach 2 is using separate Oauth 2.0 bearer token to access token introspection endpoint.

## Approach 1: Token Introspection for RestliGateway users

This approach includes adding token introspection endpoint in oauth2 authorization server.

Diagram, text

Description automatically generated

1. Client to call this endpoint by passing token parameter **client\_id** and **client\_secret** in the request post body for client authentication.
2. Implement post endpoint in oauth-frontend.
3. Partner to pass token, **client\_id**, and **client\_secret**.
4. Oauth-frontend validates the client by validating AT.
5. Implementation can open the access token and validate with passed in **client\_id** and **client\_secret** to return the metadata of the access token if validation is successful.
6. Oauth-frontend calls login-server to fetch the token metadata information.

## Benefits

1. Any valid authenticated third party application can access this endpoint with valid **client\_id** and **client\_secret**.
2. By adding this endpoint in oauth-frontend, it’s easier for partners to access all the oauth related information at one place.

**Note:** You must add basic fuse limitation to introspection endpoint to counter token phishing attack.

## Metadata

Below are the metadata of the access/refresh token that can be included in the response and is useful to partners(HSEC approval required to finalize these fields):

* Client\_Id(required field)
* PersonUrn of the user with the access token. To check with HSEC if it is safe to return this field. This is an optional field in case of 2 legged AT.
* List of scope member permissions that the user has granted for the app in this access token. Optional field in case of 2 legged AT.
* Access token expiry/TTL. This is an optional field. Currently the non-expiry token is not supported but good to have this optional for future use.
* Active (required boolean field). True if AT is active and false in case of expired/revoked. To check with HSEC if we can return status field with active/expired/revoked status.
* Access token created time(required field).
* Last authorized at(required field).

## Token Introspection Request

Make the following HTTP POST request with a Content-Type header of x-www-form-urlencoded:

**POST**

<https://www.linkedin.com/oauth/v2/introspectToken>

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Required (Yes/No)** |
| client\_id | Application client Id | Yes |
| client\_secret | Application client secret | Yes |
| token | The string value of the access token or refresh token returned from the token endpoint. | Yes |

## Sample Request

|  |  |
| --- | --- |
| POST | /oauth/v2/introspectToken HTTP/1.1 |
| Host | www.linkedin.com |
| Content-Type | application/x-www-form-urlencoded |

token={your\_token}&client\_id={your\_client\_id}&client\_secret={your\_client\_secret}

## Token Introspection Response

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

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Required (Yes/No)** |
| active | Boolean indicator of whether or not the presented token is currently active | Yes |
| status | Status of the token | No |
| scope | A JSON string containing comma separated list of scopes associated with this token | No |
| client\_id | Client identifier for the OAuth 2.0 client that requested this token | No |
| created\_at | Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token was originally issued | No |
| expires\_at | Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token will expire | No |
| authorized\_at | Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when token was authorized | No |

## Sample Responses

**Token revoked:**

curli -d 'token=AQWLmNqvW4JCbEePhbGpLhr7OesejxQC9YhxisJvIY2t6Pv8zCjGksak3NaLt4gBgU1dyV8SdX5X2fA1ebcJ7eVdvwX7Ii\_m3pSr\_2OveUf4NpAr5vnkeNJXa7av6KBn4IKQhJ7ao0YqI91g6miZ3puQBksYFoTtckOBVvH4z-T\_C4it5kcVsGTKcvZn5aQ9JdNABKTLhSeoFu6Q52cRWWaT\_jbdHU6E4jjazjNQUKOZiKA6h5pnh2ppSJcQUUwCvBiSfiA2oe205Kp3txqvYkPdkDrEHuCqlUHDixQhWcepWL3pkV4fKLLj9kNJq0X2dCRYLSuTCZXYuye6yYsDfxmwt1Dp\_Q&client\_id=84o1i5mjq59xuv&client\_secret=Yn4GaPGMZSwmG2J3' https://www.linkedin-ei.com/oauth/v2/introspectToken -v -H "Content-Type: application/x-www-form-urlencoded"

{

"active": false,

"client\_id": "84o1i5mjq59xuv",

"created\_at": 1587497291083,

"status": "revoked",

"expires\_in": 1587497620127,

"scope": "r\_basicprofile,rw\_organization,w\_share"

}

**Token expired:**

{

"active": false,

"client\_id": "84o1i5mjq59xuv",

"authorized\_at": 1587497291000,

"created\_at": 1587497291083,

"status": "expired",

"expires\_in": 1587497620127,

"scope": "r\_basicprofile,rw\_organization,w\_share"

}

**Valid token:**

{

"active": true,

"client\_id": "84o1i5mjq59xuv",

"authorized\_at": 1587497291000,

"created\_at": 1587497291083,

"status": "active",

"expires\_in": 1587497620127,

"scope": "r\_basicprofile,rw\_organization,w\_share"

}

**Passed in client information doesn’t match the token information** (If client credentials passed in the request doesn’t match with the client information in the access token)

{

"active": false

}

**Error response**

|  |  |
| --- | --- |
| **Description** | **Error response** |
| If client credentials passed in the request is not valid | HTTP 401 (Unauthorized) |
| If access token verification results in member\_restricted | HTTP 401(Unauthorized) |
| If app hits fuse throttling for this endpoint | HTTP 429 |

# References

* **<https://tools.ietf.org/html/rfc7662>**
* Implementation proposed in this document is based on the IETF RFC for Token Introspection: **<https://tools.ietf.org/html/rfc7662>**