Token Introspection

03/19/2020

# TOC:

1. Introduction
2. Problem Statement explaining core issues of users
3. Audience analysis and use cases. For example, new user scenario and advanced user scenario. Can include a visual representation of information
4. Approach or How to resolve the problem? Can include problem specific workflows
5. APIs
   1. Explain the API usage
   2. Include the steps to execute
   3. Explain the method
   4. Tabulate the parameters and examples
   5. Document the response and show samples
6. Reference materials if any

# Abstract – Start with problem statement

Given an access/refresh token, valid partners should have a way to fetch metadata information about the token. Implementation proposed in this document is based on the IETF RFC for Token Introspection: [**https://tools.ietf.org/html/rfc7662**](https://tools.ietf.org/html/rfc7662)

# Motivation

Partners often reach out partner engineers to debug failing API calls due to permissions issues as partners don’t have token introspection endpoints to find out what scopes are associated with an access token. Token introspection endpoint will also be useful if partners need to programmatically generate access token using refresh token when access token is expired, partner can check TTL to go through the refresh token flow and get a new AT, instead of trying to use an expired access token.

Currently, token introspection endpoint is available in the Partnerin tool and by exposing a similar kind of endpoint to partners will help resolve scope permissions related issues.

# Problem Statement

Partners do not have the ability to look up metadata about a token like what permissions have been granted, token TTL etc. This is a challenge when calls are failing due to permission issues and the partner doesn’t know what scopes are associated with the token they’re using. It’s also been a challenge during the permissions migration and partners don’t know which tokens are still scoped to old permissions. A token lookup API or UI tool similar to what PartnerIn has would solve this.

# Architecture/Implementation

Given an access/refresh token, return metadata information. 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.

To prevent token scanning attacks, RFC recommends to provide some form of authorization to access this endpoint, such as client authentication as described in OAuth 2.0 [[RFC6749](https://tools.ietf.org/html/rfc6749)] or a separate OAuth 2.0 access token such as the bearer token, approach 1 below is using client authentication and approach 2 is using separate Oauth 2.0 bearer token to access token introspection endpoint.

**Token Introspection for RestliGateway users**

**Finalized Approach**:

Adding token introspection endpoint in oauth2 authorization server, client can call this endpoint by passing token parameter and passing client\_id and client\_secret in the request post body for client authentication.

Implement post endpoint in oauth-frontend, partner can pass token, client\_id and client\_secret.

Oauth-frontend will validate the client by validating AT. Implementation can open the access token and validate with passed in client\_id and client\_secret and return the metadata of the access token if validation is successful. Oauth-frontend calls login-server to fetch token metadata information.

Token introspection request parameters

Token : Required field, token to be introspective

Client\_id : Required field, client id of the third party application required for client authentication.

Client\_secret: Required, client\_secret of the third party application required for client authentication.

Sample call where request posted to the token introspection endpoint with the client credentials in the request’s body.

POST [/oauth/v2/introspect](https://www.linkedin-ei.com/oauth/v2/introspect) HTTP/1.1

Host: [https://www.linkedin-ei.com](https://www.linkedin-ei.com/oauth/v2/introspect)

Accept: application/json

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

token=<token\_to\_be\_introspected>&client\_id=<client\_id>&client\_secret=<client\_secret>

Pros:

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 stuff at one place.

Note: Need to add basic fuse limitation to introspection endpoint to counter token phishing attack.

Below are the metadata of the access/refresh token that can be included in the response and will be helpful for partners(HSEC approval required to finalize these fields).

* Client\_Id(required field)
* PersonUrn of the user this access token is for(check with HSEC is it safe to return this field), this will be an optional field incase 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, optional field (We don’t support non expiry token as of now, but to good to have this optional incase if we support future)
* Active (required boolean field), true if AT is active and false incase of expired/revoked(**check with HSEC whether we can return status field with active/expired/revoked status and if there are any concerns)**.
* 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** |
| 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** | **Mandatory** |
| active | Boolean indicator of whether or not the presented token  is currently active | yes |
| status |  | 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"

}

**Passedin 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**](https://tools.ietf.org/html/rfc7662)