



OAuth for IXP Operators

35th Euro-IX Forum

Zaandam, Netherlands

20 - 22 October, 2019

Barry O'Donovan

barry.odonovan@inex.ie

[@ComePeerWithMe](#) / [@barryo79](#)



Sign Up

Already a member? [Log In](#)

 Sign up with Facebook

 Sign up with Google

or

Sign up with email

Join this site's community. [Read more](#)

Login

 Login with Facebook

 Login with Google

 Login with Instagram

 Login with Amazon

Email

Password

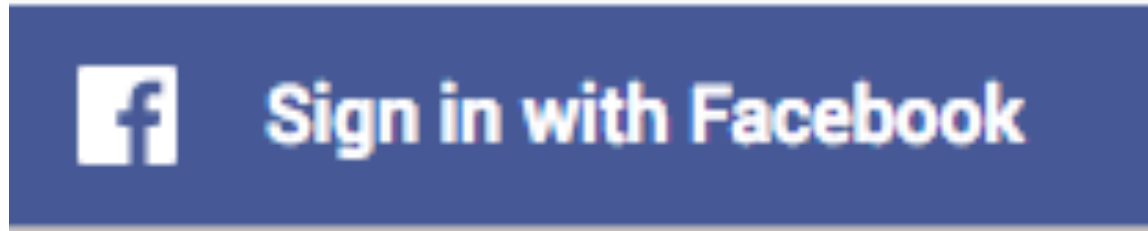
[FORGOT YOUR PASSWORD?](#)

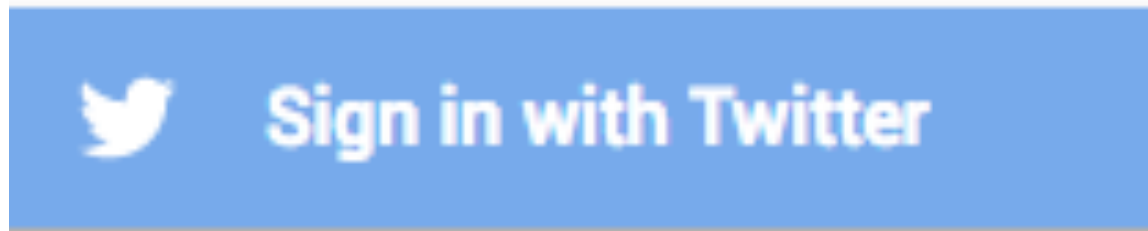
SIGN IN

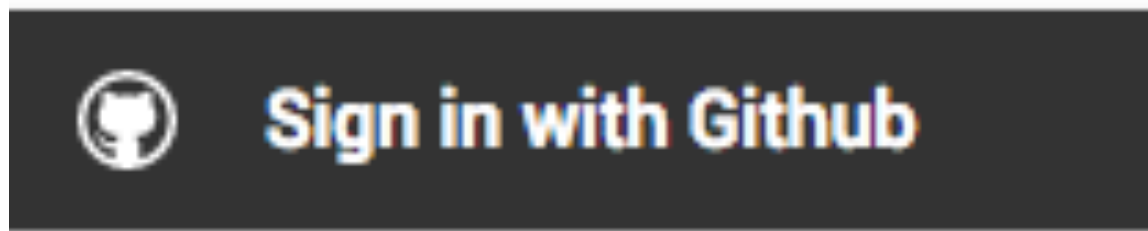
[CREATE ACCOUNT](#)

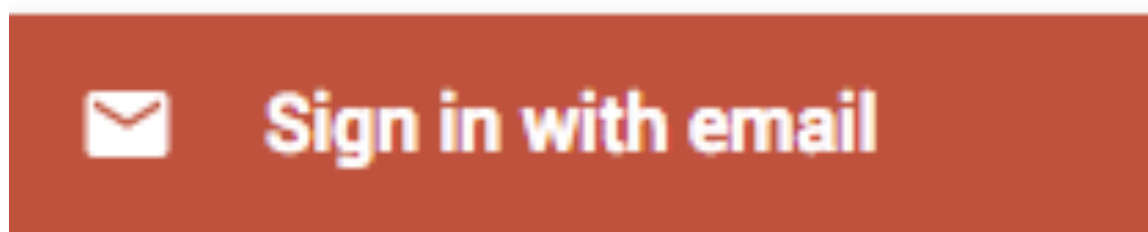
[RETURN TO STORE](#)

 Sign in with Google

 Sign in with Facebook

 Sign in with Twitter

 Sign in with Github

 Sign in with email

 Sign in with phone

An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications.



– OAuth 2.0 Definition

Why is this relevant for IXP operators?

OAuth 2.0 Roles

- The **resource owner** is the *user* - you and I.
- The **client** is the *third party application* looking for access to the *user's* account.
- The **authorization server** is that which presents the interface for the *user* to approve / deny access to the *client*.
- The **resource server** is the API server used to access the *user's* information (*often the same as the authorization server*).

OAuth 2.0 - IDs, Secrets and URLs

 **PeeringDB** [barryo](#) 

[Advanced Search](#)

Register a new application

Name

Client id

Client secret

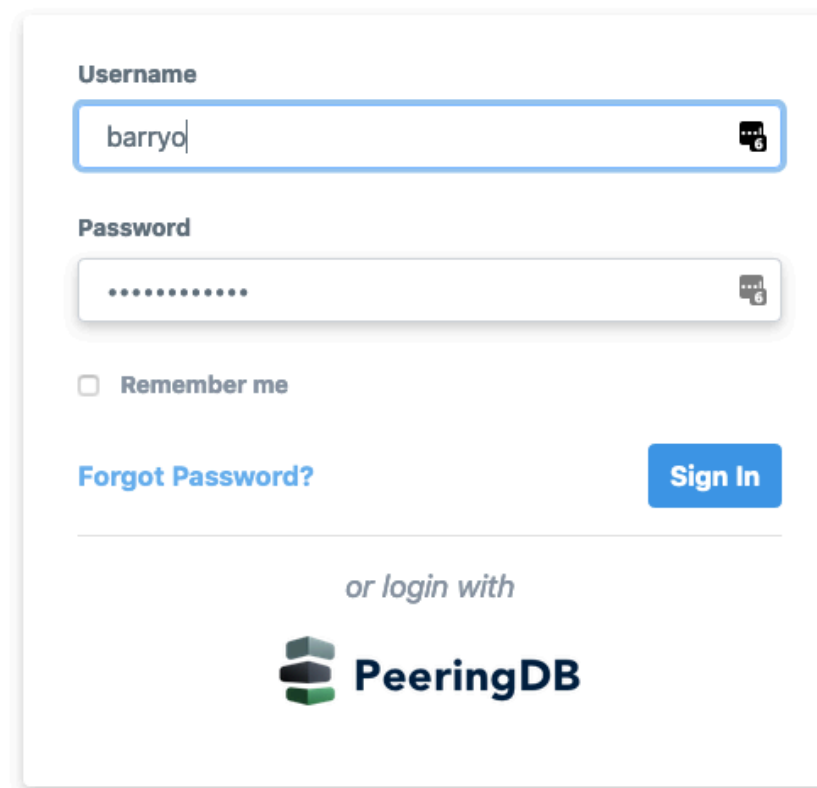
Client type

Authorization grant type


Redirect uris

Example OAuth Authorization Process

Let's look at IXP Manager with PeeringDB.



The screenshot shows a login form with the following elements:

- Username:** A text input field containing the text "barryo".
- Password:** A password input field with masked characters (dots).
- Remember me**
- [Forgot Password?](#)
- Sign In** button
- Text: *or login with*
-  **PeeringDB**

- What happens if we click on *Login with PeeringDB*?

Example OAuth Authorization Process

User clicks on *Login with PeeringDB* [1]:

1. HTTP GET request to *client* [2]: `/auth/login/peeringdb`

2. Returns a HTTP redirect response to send the *user* to [3]:

```
https://auth.peeringdb.com/oauth2/authorize/  
?response_type=code  
&client_id=CLIENT_ID  
&redirect_uri=REDIRECT_URI  
&scope=profile+email+networks  
&state=1234zyx
```



User

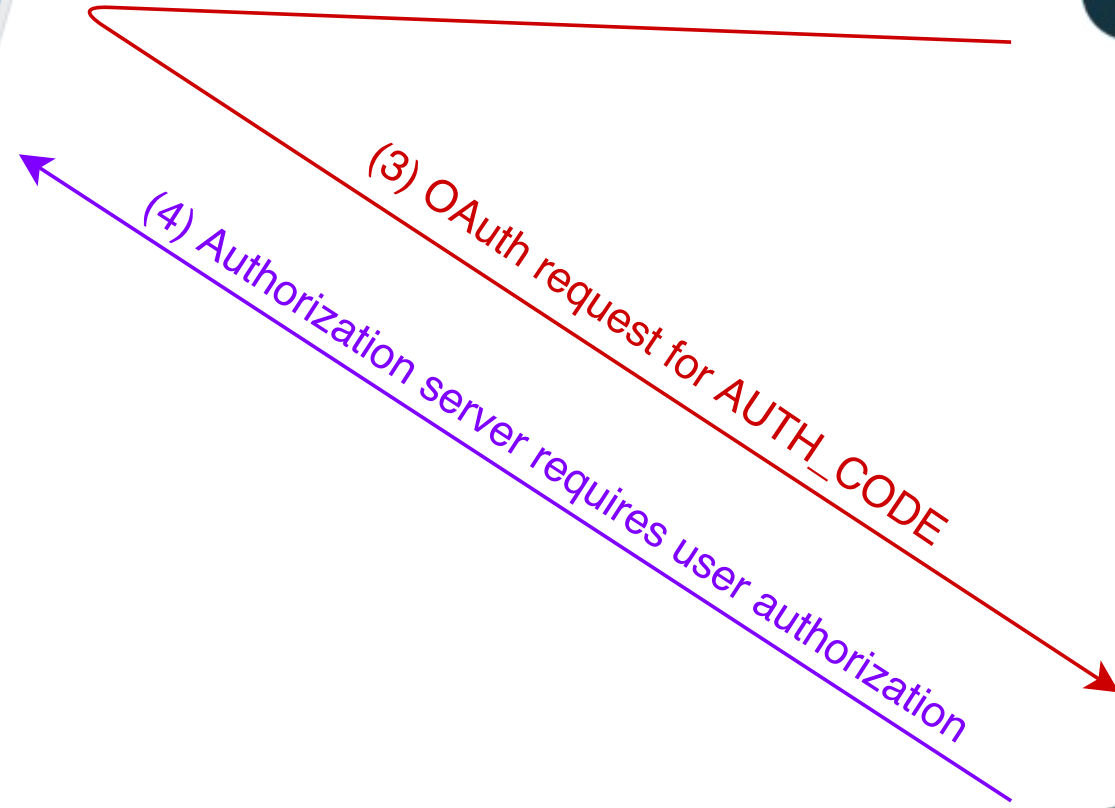
(1) Clicks "Log in with..."



(2) HTTP GET Request for OAuth Process



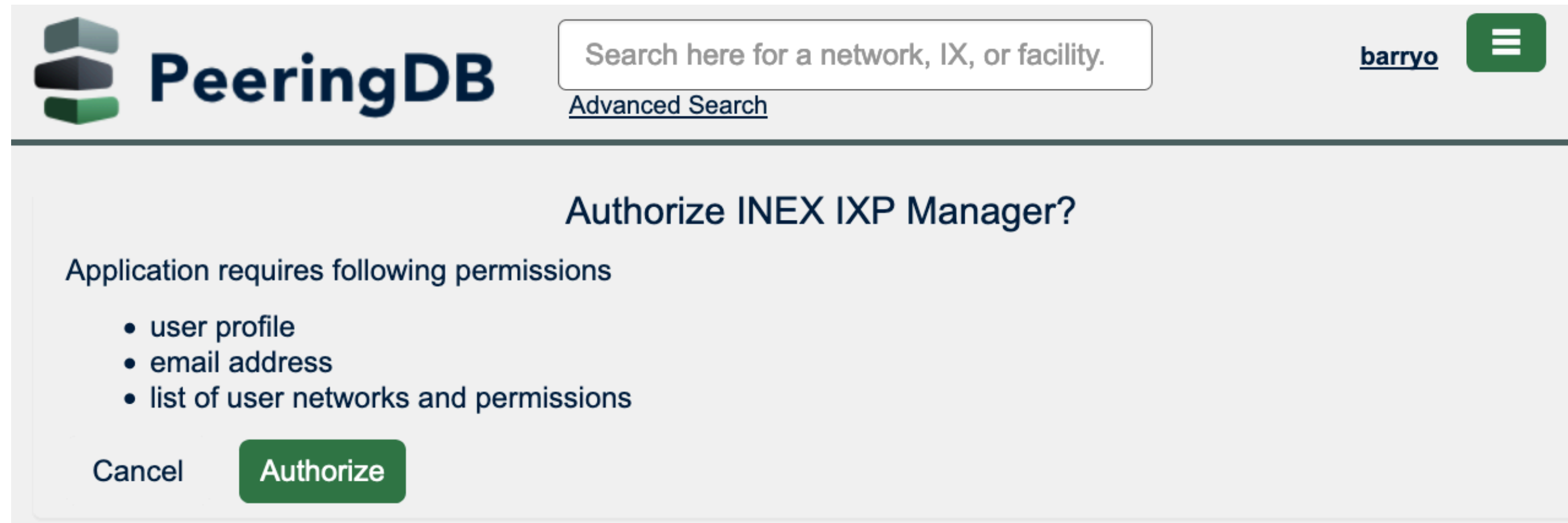
Client



Authorization (and resource) server



Example OAuth Authorization Process



The screenshot shows the PeeringDB website interface. At the top left is the PeeringDB logo. To its right is a search bar with the placeholder text "Search here for a network, IX, or facility." and a link for "Advanced Search". On the top right, the user's name "barryo" is displayed next to a green menu icon. The main content area is titled "Authorize INEX IXP Manager?". Below the title, it states "Application requires following permissions" and lists three permissions: "user profile", "email address", and "list of user networks and permissions". At the bottom of the authorization box, there are two buttons: "Cancel" and "Authorize".

Asked to authorize **INEX's** IXP Manager [4].
(*And note the requested scopes*)

Example OAuth Authorization Process

If the *user* clicks authorize [5], the authorization service redirects back via the (verified) redirect URL [6] with an authorization code:

```
https://www.someix-ixpmanager/auth/login/peeringdb/callback  
    ?code=AUTH_CODE  
    &state=1234zyx
```

1. Use of SSL mandatory.
2. Redirect URL must match what was registered for the *client*.
3. Client must compare received state to what was sent.



User

(1) Clicks "Log in with..."
(5) Users authorizes access



(2) HTTP GET Request for OAuth Process



Client

(3) OAuth request for AUTH_CODE
(6) HTTP Redirect back to client application
(4) Authorization server requires user authorization



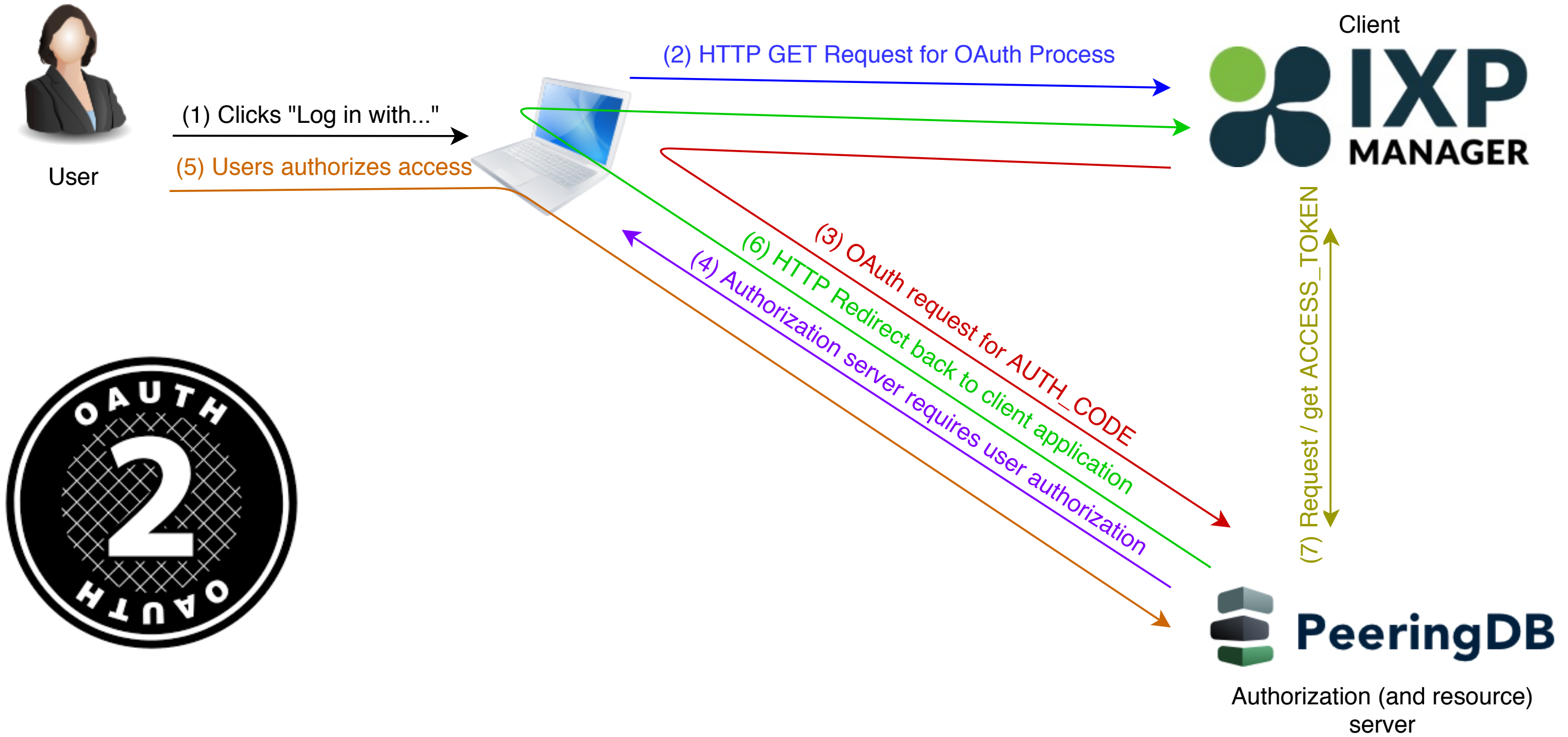
Authorization (and resource) server



Example OAuth Authorization Process

In the background, the *client* now uses the `code=AUTH_CODE` received to get an access token via a POST request to the *authorization server* [7].

```
https://auth.peeringdb.com/oauth2/token/  
?grant_type=authorization_code  
&code=AUTH_CODE  
&redirect_uri=REDIRECT_URI  
&client_id=CLIENT_ID  
&client_secret=CLIENT_SECRET
```

Example OAuth Authorization Process

Once the *client* has an *access token*, it can request *user* information with the *scope(s)* that it has been authorized for via HTTP GET [8].

```
https://auth.peeringdb.com/profile/v1
```

HTTP Headers:

```
Authorization: Bearer ACCESS_TOKEN
```



User

(1) Clicks "Log in with..."
(5) Users authorizes access



(2) HTTP GET Request for OAuth Process



Client

(3) OAuth request for AUTH_CODE

(6) HTTP Redirect back to client application

(4) Authorization server requires user authorization

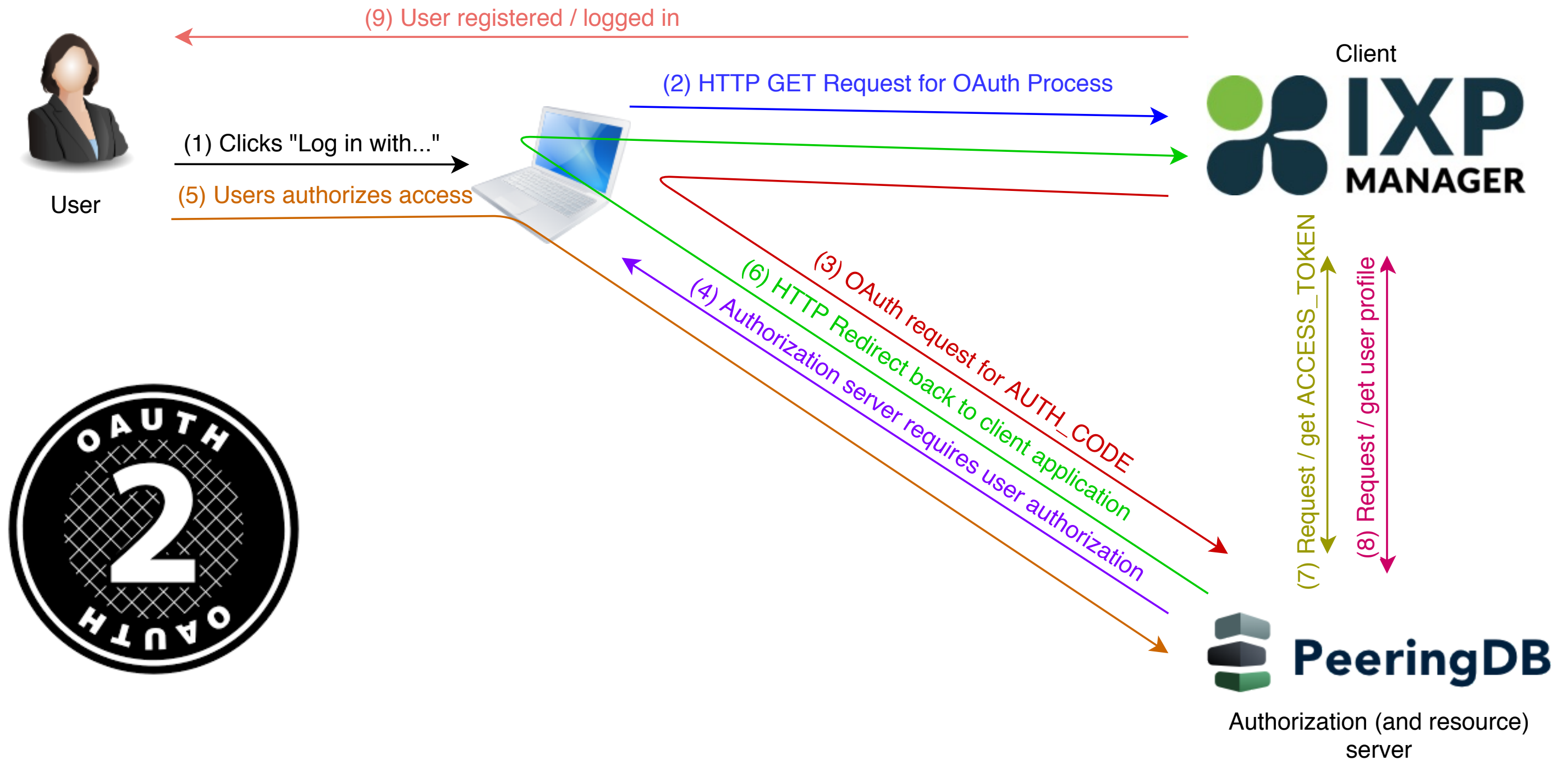
(7) Request / get ACCESS_TOKEN

(8) Request / get user profile



Authorization (and resource) server





Example OAuth Authorization Process

Remember, from a *user* perspective, this is usually two clicks.

1. **Click *Login with PeeringDB*** [1]

- browser gets redirected to PeeringDB asking for *user* permission [2,3,4].

2. **Grant permission** [5]

- browser gets redirected back to client from authorization server [6]
- client receives AUTH_CODE which is exchanged for an ACCESS_TOKEN [6,7]
- client uses ACCESS_TOKEN to get user information [8]
- client creates and/or logs user in

3. User logged into client application. [9]

Sample User Profile from PeeringDB

```
{
  "id": 9999,
  "name": "Barry O'Donovan",
  "given_name": "Barry",
  "family_name": "O'Donovan",
  "email": "barry.odonovan@inex.ie",
  "verified_user": true,
  "verified_email": true,
  "networks": [
    {
      "perms": 15, "asn": 65500, "name": "Acme Net", "id": 9999
    }, {
      "perms": 15, "asn": 65501, "name": "Example Net", "id": 9998
    }
  ]
}
```

IXP Manager Verification (1/2)

How does IXP Manager validate & use user detail from PeeringDB?

- data structure okay (user details present, network(s) present)?
- user has `verified_user` and `verified_email` with PeeringDB?
- at least one of the networks are IX members?
- load (by PeeringDB ID) or create user object in IXP Manager
- created user is a read-only user by default

IXP Manager Verification (2/2)

- remove any user/network associations in IXP Manager that previously came from PeeringDB but are no longer present in the new PeeringDB network list
- add any new user/network associations (only if a *normal peering network* that is current, connected and hasn't requested PeeringDB OAuth be disabled for them)

Then either:

- if no user/network associations at end of process, delete user;
- otherwise log user in.

Do We Trust PeeringDB?

So Do We Trust PeeringDB?

This is a reasonably small industry where the significant human actors are well known.

So yes, we trust PeeringDB 😊

What Are the Risks?

1. OAuth protocol is well understood, widely used and sound.
2. IXP Manager and PeeringDB use well established libraries for OAuth server / client.
3. Implementation issues?

What's the Exposure

To my mind, not a lot:

- Port details, IP addressing, NOC details (available via IX-F Export, PeeringDB, IX website)
- Traffic graphs, peer to peer graphs
- Again, read-only access by default
- Again, absolutely no superadmin access via OAuth

INEX's Experience with PeeringDB OAuth

- Launched August 29th, 2019
- 26 new users created since
 - 19 via PeeringDB, 2 by member admins, 5 by ops team
 - i.e. 65% of new users required no other actor
- Feedback has been 100% positive
 - no member has requested an opt-out

IXP Manager Support

- Released in IXP Manager v5.2.0 on September 20th
- Enabling PeeringDB OAuth is really easy¹:
 1. Register your IXP Manager instance as an [OAuth application on PeeringDB](#).
 2. Add configuration elements to `.env`:

```
AUTH_PEERINGDB_ENABLED=true  
PEERINGDB_OAUTH_CLIENT_ID="xxx"  
PEERINGDB_OAUTH_CLIENT_SECRET="xxx"  
PEERINGDB_OAUTH_REDIRECT="https://www.my-ixpmanager-url.com/auth/login/peeringdb/callback"
```

¹ <https://docs.ixpmanager.org/features/peeringdb-oauth/>

References

- [IXP Manager documentation for enabling PeeringDB OAuth](#)
- [PeeringDB OAuth 2.0 Documentation](#)
- [OAuth 2.0 Community Site, rfc6749, rfc6750, rfc6819](#)
- [OAuth 2 Simplified](#) - excellent blog post.
- [Laravel Socialite](#) and [Laravel Passport](#) (via [oauth2-server](#))
- [Python Django OAuth Toolkit](#) (via [OAuthLib](#))



THANK YOU

Any Questions?

