INEX

OAuth with PeeringDB

For Network Operators

UKNOF45, London, UK.

January 15th, 2020.

Barry O’Donovan

@ComePeerWithMe / @barryo79

https://www.inex.ie/

INEX

- Peering point for the island of Ireland, member owned association, not for profit, founded in 1996
- ~100 members
- Peak of ~400Gbps
- Dual infrastructure, 8 PoPs, own dark fibre
- Opened INEX Cork in 2016
- Home of IXP Manager

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*

INEX

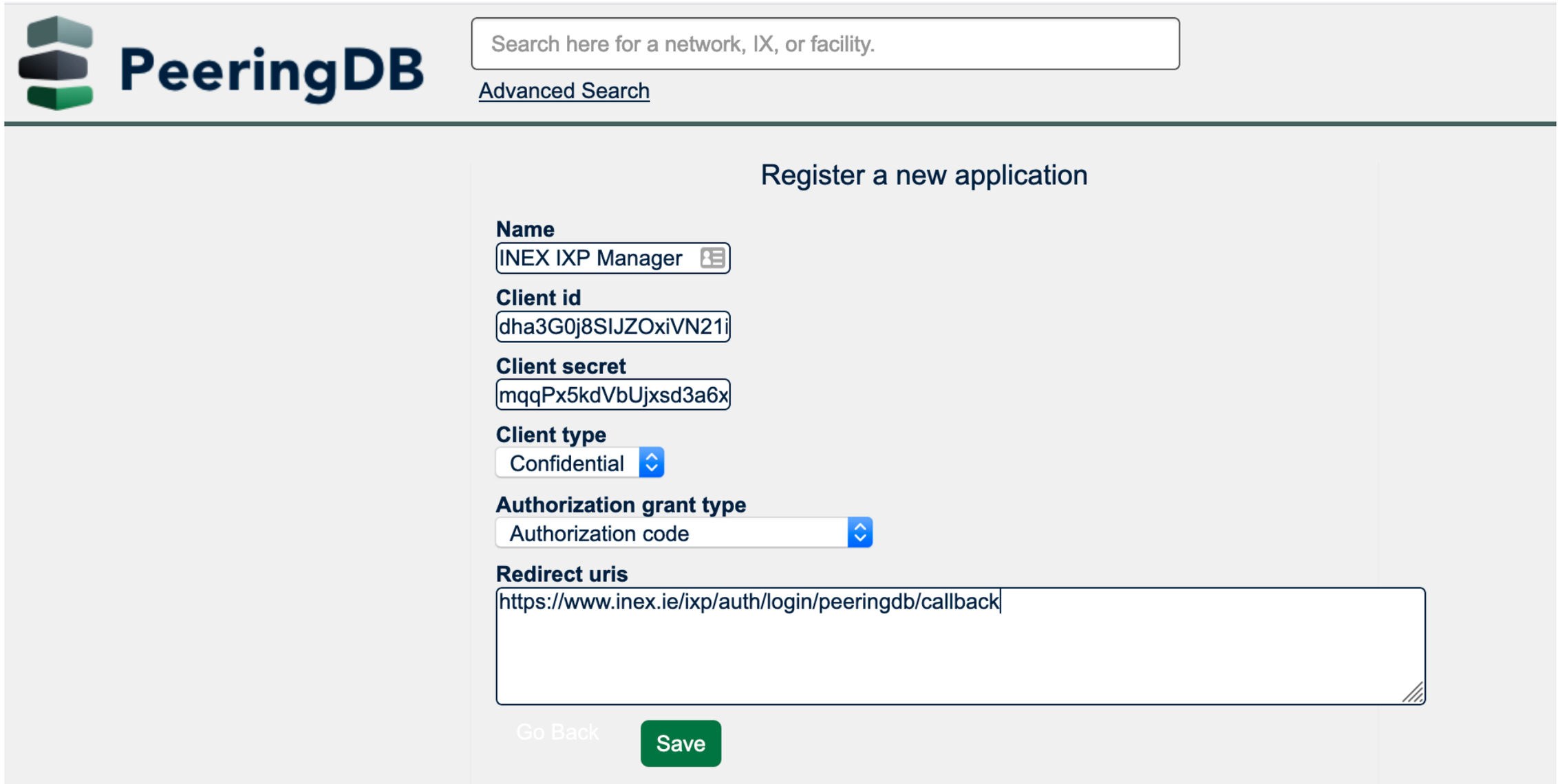
Why is this relevant for network operators?

OAuth 2.0 Roles

- The **resource owner** is the *end-user* (for us at least).
- 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)*.

INEX

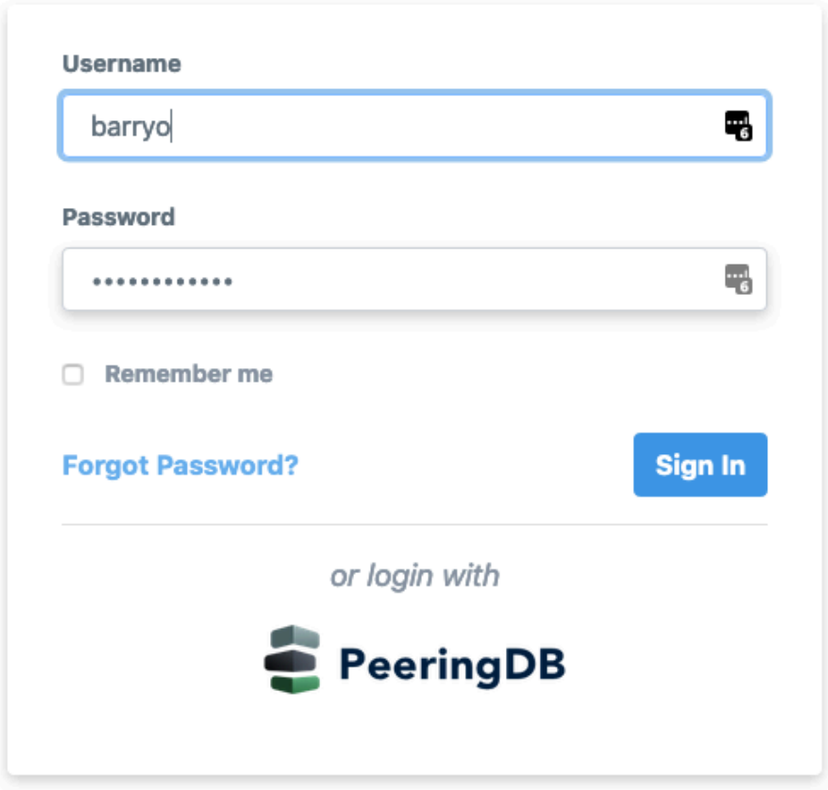
OAuth 2.0 - IDs, Secrets and URLs



INEX

Example OAuth Process

Let's look at IXP Manager with PeeringDB.



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

INEX

Example OAuth 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
```

INEX

User

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

(2) HTTP GET Request for OAuth Process

Client

IXP MANAGER

(3) OAuth request for AUTH_CODE

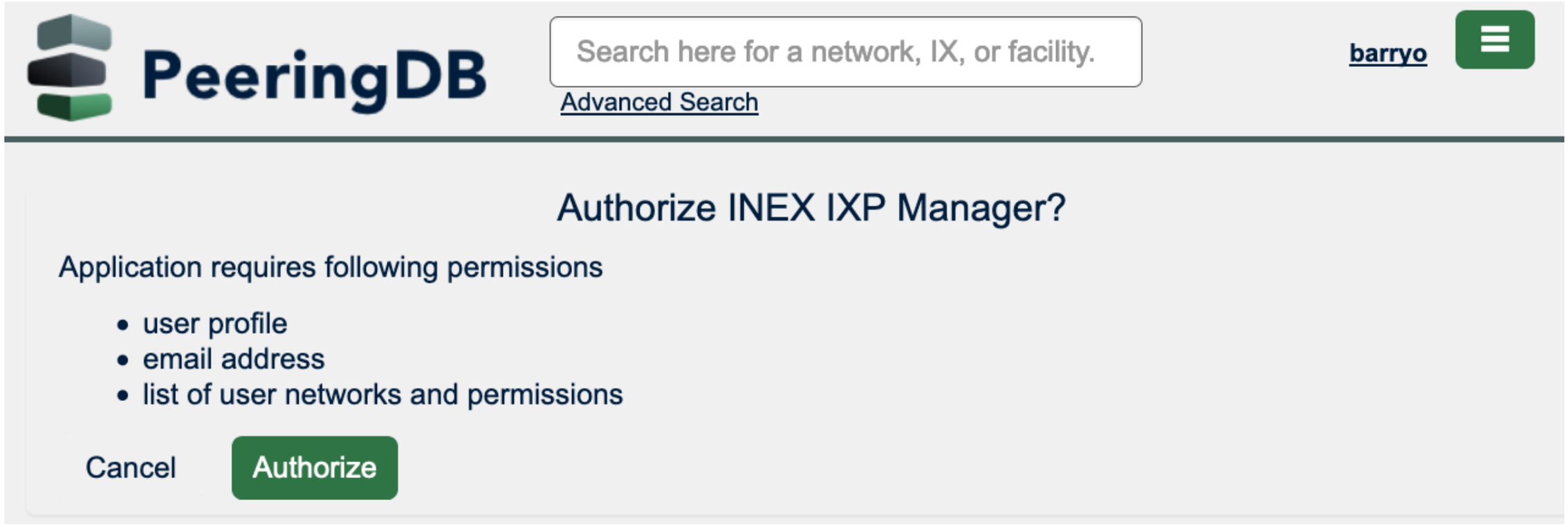
(4) Authorization server requires user authorization

PeeringDB

Authorization (and resource) server

INEX

Example OAuth Process



Asked to authorize **INEX's** IXP Manager [4].

(And note the requested scopes)

INEX

Example OAuth 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
```

Note that (a) use of TLS mandatory; (b) redirect URL must match what was registered for the *client*; and (c) client must compare received state to what was sent.

INEX

User

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

(5) Users authorizes access

(2) HTTP GET Request for OAuth Process

(3) OAuth request for AUTH_CODE

(6) HTTP Redirect back to client application

(4) Authorization server requires user authorization

Client

IXP MANAGER

PeeringDB

Authorization (and resource) server

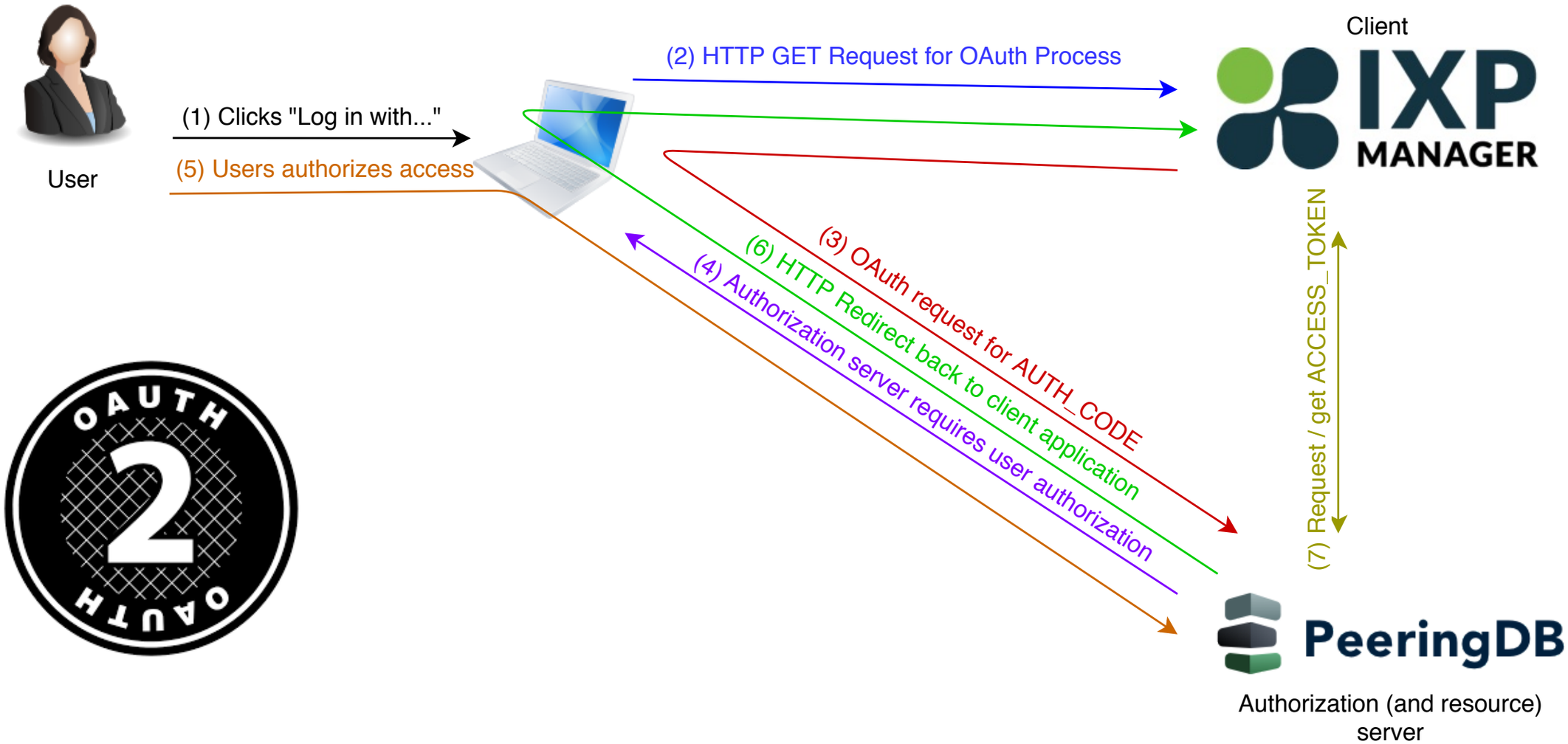
INEX

Example OAuth 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
```

INEX


User
(1) Clicks "Log in with..."
(5) Users authorizes access
(2) HTTP GET Request for OAuth Process
(3) OAuth request for AUTH_CODE
(6) HTTP Redirect back to client application
(4) Authorization server requires user authorization
Client
IXP MANAGER
(7) Request / get ACCESS_TOKEN
PeeringDB
Authorization (and resource) server

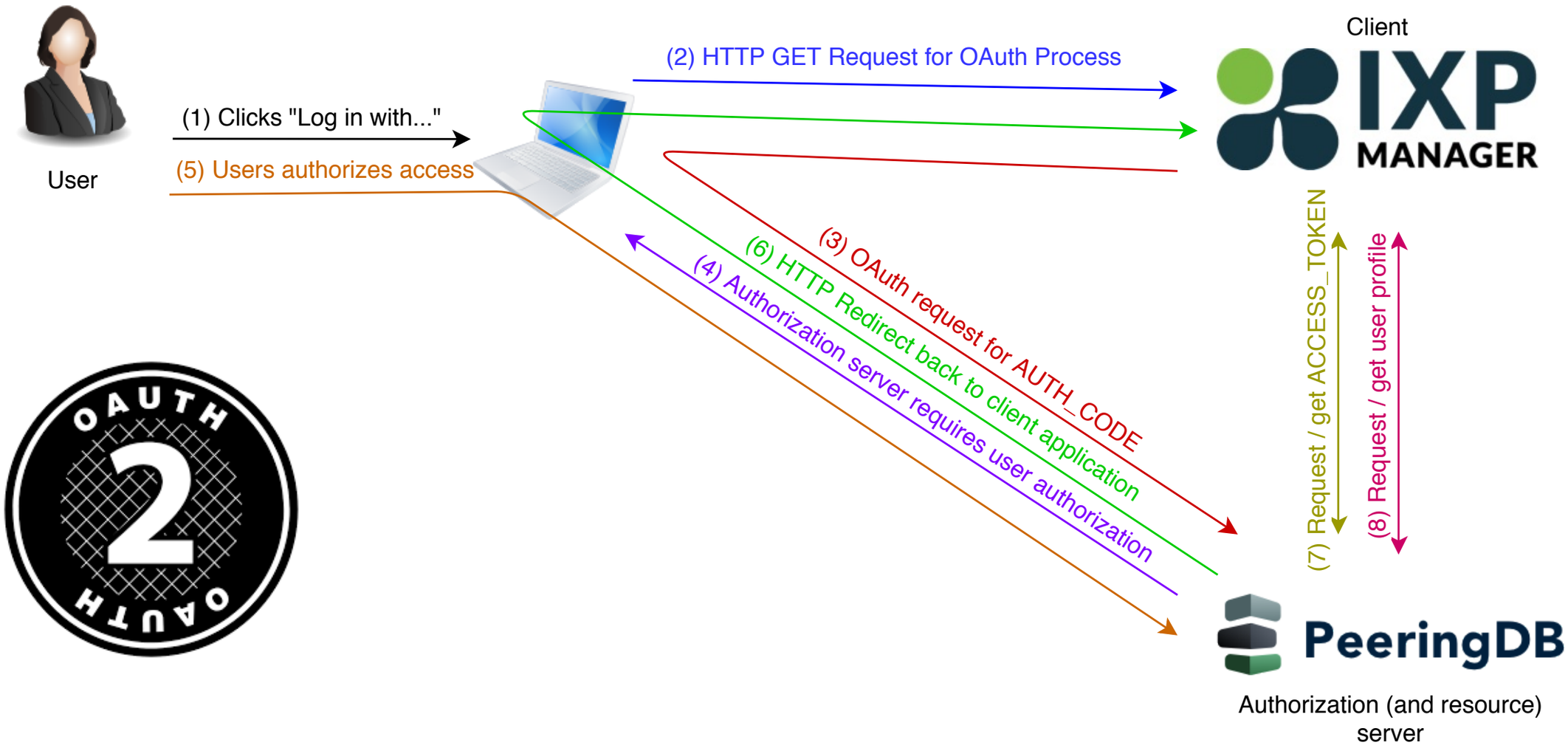
INEX

Example OAuth Process

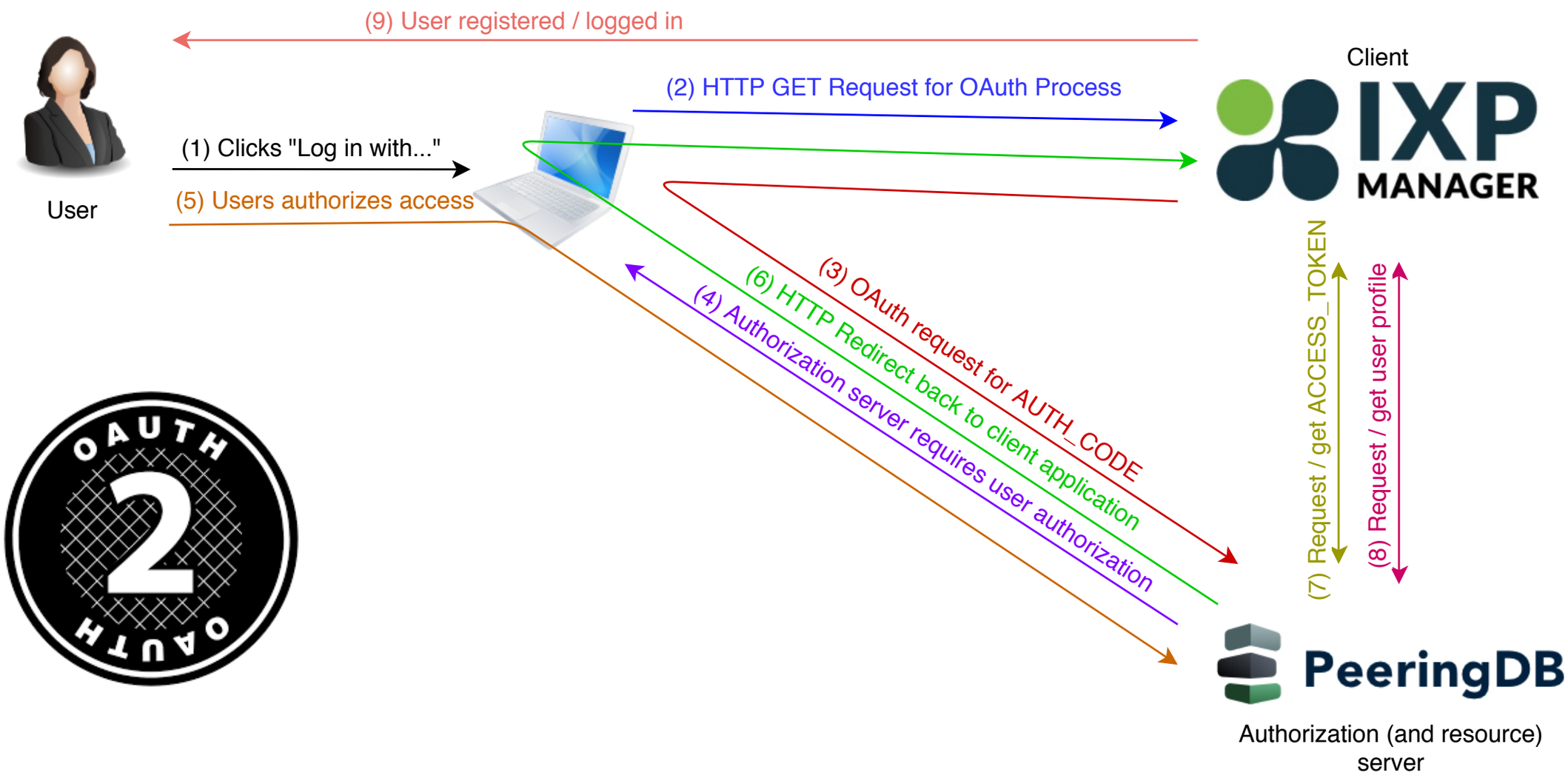
Once the *client* has an *access token*, it can request *user* information with the *scope(s)* that it has been authorized for via HTTP GET [8].

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

```
HTTP Headers:
    Authorization: Bearer ACCESS_TOKEN
```


Client
IXP MANAGER
User
(1) Clicks "Log in with..."
(2) HTTP GET Request for OAuth Process
(3) OAuth request for AUTH_CODE
(4) Authorization server requires user authorization
(5) Users authorizes access
(6) HTTP Redirect back to client application
(7) Request / get ACCESS_TOKEN
(8) Request / get user profile
PeeringDB
Authorization (and resource) server

INEX


(9) User registered / logged in
Client
(2) HTTP GET Request for OAuth Process
(1) Clicks "Log in with..."
(5) Users authorizes access
User
IXP MANAGER
(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
PeeringDB
Authorization (and resource) server

OAUTH
2
OAUTH

INEX

Example OAuth 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 exchanges 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]

INEX

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
    }
  ]
}
```


INEX

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

INEX

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.

INEX

Do We Trust PeeringDB?

INEX

So Do We Trust PeeringDB?

This is a reasonably small industry where the significant human actors are well known to many of us.

So yes, we trust PeeringDB 😄

(evaluate your own security/threat model!)

INEX

What Are the Risks?

1. OAuth protocol is well understood, widely used and proven.

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

INEX's Experience with PeeringDB OAuth

- Launched August 29th, 2019
- 26 new users created in first two months:
  - 19 via PeeringDB, 2 by member admins, 5 by ops team
  - i.e. 73% of new users required no other actor
- Feedback has been 100% positive
  - no member has requested an opt-out
- Found issue with mailing list subscriptions.

INEX

IXP Manager Support

- Released in IXP Manager v5.2.0 on September 20th
- Enabling PeeringDB OAuth is really easy[1]:

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"
```

[1] https://docs.ixpmanager.org/features/peeringdb-oauth/

References

- IXP Manager PeeringDB OAuth Documentation
- 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)

INEX

Thank You!

@ComePeerWithMe - @barryo79

https://www.inex.ie/

https://www.ixpmanager.org/

INEX