Setting up OpenID Connect for a New Application

Contents
- Overview
- Requesting OpenID Connect/OAuth 2.0 Service
- Authenticating the user
  1. Create an anti-forgery state token (optional but recommended)
  2. Send an authentication request to NYU Identity Server
  3. Confirm anti-forgery state token (optional but recommended)
  4. Exchange code for access token and ID token
  5. Obtain user information by using access_token
- Production and Sandbox Logout URLs
- Other Documents
- Contact Information

Overview

NYU's OAuth 2.0 APIs can be used for both authentication and authorization. This document describes our OAuth 2.0 implementation for authentication.

OAuth 2.0 (also known as OAuth2) provides a quick and easy way to have your application authenticate users with their NYU NetID without having to install any client software in your application. The process is simply handled over a short series of HTTP GET and POST methods once you register your application with the NYU IT Identity Services team.

Requesting OpenID Connect/OAuth 2.0 Service

Before your application can use NYU's OAuth 2.0 authentication system for user login, you must register your application with us by filling out the OpenID Connect Planning Questionnaire.¹

- Developer name, email address, and phone.
- Department, unit, or college code.
- Application name (10 characters or less).
- Redirect URL to your application where you want the NYU OAuth 2.0 authorization endpoint to send users once they've authenticated. Please note that this service requires the use of HTTPS.

¹ OpenId Connect Integration Planning Questionnaire - Information we require from requester: App owner/tech team info and most importantly, the application Redirect URL (Also referred to as the Callback URL. Must be https://); See Other Documents section of this guide
Authenticating the user

The OpenID Connect protocol, in abstract, follows the following steps.

1. The Relying Party (Client) sends a request to the OpenID Provider (OP).
2. The Relying Party authenticates the End-User and obtains authorization.
3. The Relying Party responds with an ID Token and usually an Access Token.
4. The Relying Party can send a request with the Access Token to the UserInfo Endpoint.
5. The UserInfo Endpoint returns Claims about the End-User. These steps are illustrated in the following diagram:

![Diagram](image)

Authenticating the user involves obtaining an ID token and validating it. ID tokens are a standardized feature of OpenID Connect designed for use in sharing identity assertions on the Internet.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Production</th>
<th>Sandbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization Endpoint URL:</td>
<td><a href="https://auth.nyu.edu/oauth2/authorize">https://auth.nyu.edu/oauth2/authorize</a></td>
<td><a href="https://qaauth.home.nyu.edu/oauth2/authorize">https://qaauth.home.nyu.edu/oauth2/authorize</a></td>
</tr>
<tr>
<td>Token Endpoint URL:</td>
<td><a href="https://auth.nyu.edu/oauth2/token">https://auth.nyu.edu/oauth2/token</a></td>
<td><a href="https://qaauth.home.nyu.edu/oauth2/token">https://qaauth.home.nyu.edu/oauth2/token</a></td>
</tr>
<tr>
<td>User Info Endpoint URL:</td>
<td><a href="https://auth.nyu.edu/oauth2/userinfo">https://auth.nyu.edu/oauth2/userinfo</a></td>
<td><a href="https://qaauth.home.nyu.edu/oauth2/userinfo">https://qaauth.home.nyu.edu/oauth2/userinfo</a></td>
</tr>
</tbody>
</table>

1. Create an anti-forgery state token (optional but recommended)

You must protect the security of your users by preventing request forgery attacks. The first step is creating a unique session token that holds state between your app and the user's client. You later match this unique session token with the authentication response returned by the NYU OAuth Login service to verify that the user is making the request and not a malicious attacker. These tokens are often referred to as cross-site request forgery (CSRF) tokens.
The following code demonstrates generating unique session tokens in PHP.

```php
// Create a state token to prevent request forgery. 
// Store it in the session for later validation. 
$state = sha1(openssl_random_pseudo_bytes(1024));
$app['session']->set('state', $state);

// Set the client ID, token state, and application name in the HTML while 
// serving it.
return $app['twig']->render('index.html', array( 
    'CLIENT_ID' => CLIENT_ID,
    'STATE' => $state,
    'APPLICATION_NAME' => APPLICATION_NAME
));
```

2. Send an authentication request to NYU Identity Server

The next step is forming an HTTPS GET request with the appropriate URI parameters. Note the use of HTTPS rather than HTTP in all the steps of this process.

Authorization Endpoint URL: https://auth.nyu.edu:443/oauth2/authorize

The following table gives more complete descriptions of the parameters accepted by NYU's OAuth 2.0 authentication API.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Req'd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>Yes</td>
<td>The scope value must begin with the string openid and then include profile or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>email or both.</td>
</tr>
<tr>
<td>response_type</td>
<td>Yes</td>
<td>Instructs the NYU OAuth 2.0 authorization endpoint to provide an authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>code to your request. This should always be set to &quot;code&quot;.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>Yes</td>
<td>Also referred to as the Callback URL. Must be https://</td>
</tr>
<tr>
<td>client_id</td>
<td>Yes</td>
<td>The client_id provided to you by IdM Team that identifies your application.</td>
</tr>
<tr>
<td>state</td>
<td>No</td>
<td>Recommended. Unique value used to maintain state between the request and the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>callback.</td>
</tr>
</tbody>
</table>

An example URL request:

https://auth.nyu.edu:443/oauth2/authorize?
scope=openid&
state=af0iflsdkjd&
response_type=code&
redirect_uri=http%3A%2F%2Fapp1.idmdev.com%3A8080%2Foauth2client&
client_id=XASHCsdVdYDOELGm_ZB6bkJdYu8a
Once users are authenticated, they are required to give consent if your app requests any new information about them, or if your app requests account access that they have not previously approved.

3. Confirm anti-forgery state token (optional but recommended)

The response is sent to the redirect_uri that you specified in the request. All responses are returned in the query string, as shown below:

http://app1.idmdev.com:8080/playground/oauth2client?state=af0ifjsldkjd&code=e836906123a7eb5dfcf53e39fb729eb2

On the server, you must confirm that the state received from Google matches the session token you created in Step 1. This round-trip verification helps to ensure that the user, not a malicious script, is making the request.

The following code demonstrates confirming the session tokens that you created in Step 1:

```php
// Ensure that there is no request forgery going on, and that the user
// sending us this connect request is the user that was supposed to.
if ($request->get('state') != ($app['session']->get('state'))) {
    return new Response('Invalid state parameter', 401);
}
```

4. Exchange code for access token and ID token

The response includes a code parameter, a one-time authorization code that your server can exchange for an access token and ID token. Your server makes this exchange by sending an HTTPS POST request. The POST request is sent to the token endpoint - https://auth.nyu.edu:443/oauth2/token.

```bash
curl -v X POST -basic -u
-u
-H "Content-Type:application/x-www-form-urlencoded; charset=UTF-8" -k
-d "grant_type=authorization_code&
    code={code} &
    redirect_uri="https://app.nyu.edu/redirect"
https://auth.nyu.edu:443/oauth2/token
```
The request must include the following parameters in the POST body:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>The authorization code that is returned from the initial request.</td>
</tr>
<tr>
<td>client_id</td>
<td>The client ID that you obtain from the IdM Service Team</td>
</tr>
<tr>
<td>client_secret</td>
<td>The client secret that you obtain from the IdM Service Team</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The URI that you specify in the request form.</td>
</tr>
<tr>
<td>grant_type</td>
<td>This field must contain a value of authorization_code, as defined in the OAuth 2.0 specification.</td>
</tr>
</tbody>
</table>

The actual request might look like the following example:

```plaintext
POST /oauth2/v3/token HTTP/1.1
Host: auth.nyu.edu
Content-Type: application/x-www-form-urlencoded

code=4/P7q7W91a-oMsCeLvIaQm6bTrgtp7&
client_id={client_id}&
client_secret={client_secret}&
redirect_uri=https://app.nyu.edu/redirect
grant_type=authorization_code
```

A successful response to this request contains the following fields in a JSON array:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>A token that can be sent to a Google API.</td>
</tr>
<tr>
<td>expires_in</td>
<td>The remaining lifetime of the access token.</td>
</tr>
<tr>
<td>token_type</td>
<td>Identifies the type of token returned. At this time, this field always has the value Bearer.</td>
</tr>
<tr>
<td>refresh_token</td>
<td>A refresh token provides your app continuous access to NYU APIs while the user is not logged into your application</td>
</tr>
</tbody>
</table>

JSON EXAMPLE

```json
{
   "token_type":"bearer","expires_in":3299,
   "refresh_token":"9db24ade1997443625d432b138bda82",
   "access_token":"56a84e82a4b528e2dd15a119aec9a56"
}
```
5. Obtain user information by using access_token

Once you have the access token from the previous step, it can be used for retrieving user information such as email, id and names.

```
curl -k -H "Authorization: Bearer e91961906e1196cfca7eef22101c3d95"
    https://auth.nyu.edu:443/oauth2/userinfo?schema=openid
```

Production and Sandbox Logout URLs

Production:
```
https://auth.nyu.edu/commonauth?commonAuthLogout=true&type=oidc2&commonAuthCalle
rPath=https://shibboleth.nyu.edu/idp/profile/Logout
```

QA:
```
https://qaauth.home.nyu.edu/commonauth?commonAuthLogout=true&type=oidc2&common
AuthCallerPath=https://shibbolethqa.es.its.nyu.edu/idp/profile/Logout
```

Other Documents

- OIDC Integration Planning Questionnaire
- Operating Level Agreement template for NYU Login Service

Contact Information

All email communications with the IdP-Team (part of the NYU IT Identity & Database Services group) should be sent to idm.services@nyu.edu