Authentication

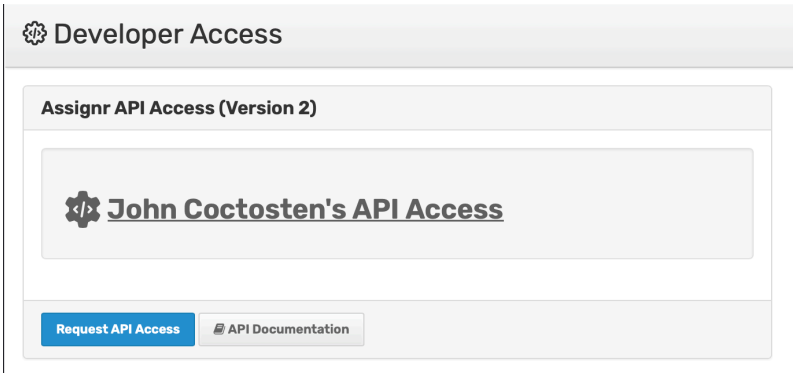
The Assignr API uses OAuth 2.0 for authentication. OAuth 2 is an authorization protocol that gives an API client limited access to user data on a web server. GitHub, Google, and Facebook APIs notably use it. OAuth relies on authentication scenarios called flows, which allow the resource owner (user) to share the protected content from the resource server without sharing their credentials. For that purpose, an OAuth 2.0 server issues access tokens that the client applications can use to access protected resources on behalf of the resource owner. For more information about OAuth 2.0, see oauth.net and RFC 6749.

For organizations that will be using the Assignr API, we support the Client Credentials flow and the Authorization Code flow. The Client Credentials flow is used when the application is only asking for access to its own data. The Authorization Code flow is used when requesting access to other user's data.

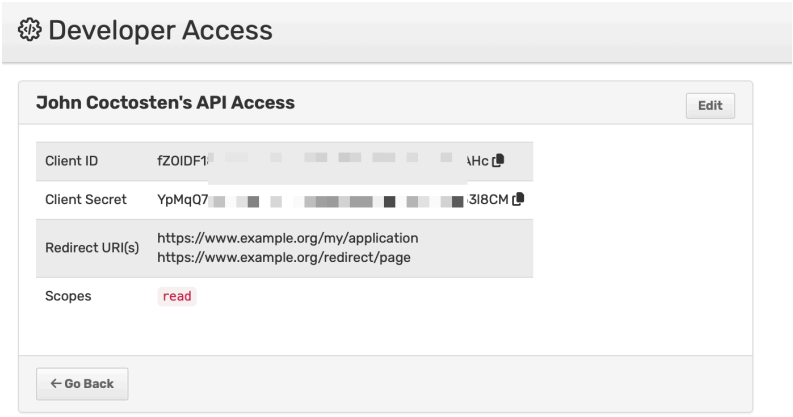
Once you have been granted access to the Assignr API, you can find your account credentials:

- Click your name, upper right corner of the screen, then choose Profile
- Choose API Access

You will then have a list of the OAuth Applications associated with your account:



The detail page will provide your Client ID and Client Secret.



Scopes

Assignr currently uses three scopes: `read`, `write` and `bank`. `write` scope is needed for any endpoint that can change data, and `bank` scope is needed for access to add/remove bank accounts or a W9.

Client Credentials Flow

Client Credentials flow is used when an application only needs access to its own data.

Shell

```
curl --location --request POST
'https://app.assignr.com/oauth/token' \
--form 'client_id="[your client id here]"'
\
--form 'client_secret="[your client secret
here]"' \
--form 'scope="read write"' \
```

```
--form 'grant_type="client_credentials"'
```

Assignr will respond with a time limited access token:

JSON

```
{
    "access_token":
"b445409253438a096fc2990740e4f50ee8e67fb2210
    "token_type": "Bearer",
    "expires_in": 7199,
    "scope": "read",
    "created_at": 1606420331
}
```

The given access token can now be used in the header of an API request.

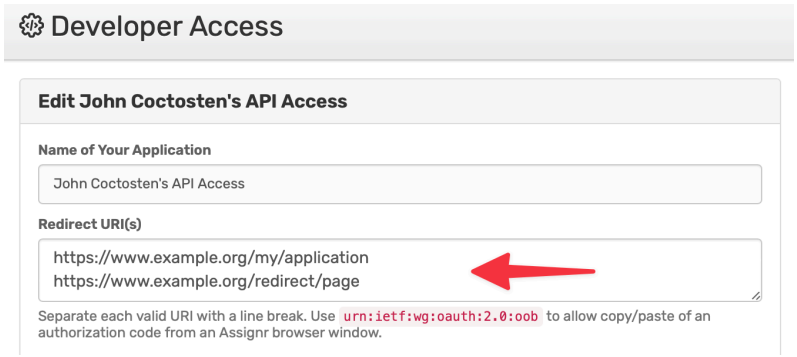
cURL

```
curl --location --request GET
'https://api.assignr.com/api/v2/sites/7/game
\
--header 'Accept:
application/vnd.assignr.v2.hal+json' \
--header 'Content-Type:
application/vnd.assignr.v2.hal+json' \
--header 'Authorization: Bearer
b445409253438a096fc2990740e4f50ee8e67fb22105
```

Authorization Code Flow

Authorization Code flow is used whenever an application is asking for a user's permission to access their data. This flow is similar to how users sign up into a web application using their Facebook or Google account. This is the approach that will be used for third party applications that need access to other users' data on the Assignr platform.

To get started, Assignr will provide you with a client ID, client secret, and the granted scopes. The redirect URI must be registered in advance within your application configuration on the Assignr website,



Obtain user authorization for your application

To request authorization from an Assignr user, your application will need to redirect the user to a specific URL:

Text

```
https://app.assignr.com/oauth/authorize?
client_id=[client_id] \
  &redirect_uri=[encoded-redirect-uri] \
  &response_type=code \
  &scope=read+write
```

The user will see a screen which requests that they authorize your application to retrieve data on their behalf.

Once they accept your authorization request, Assignr will redirect the user to the URL you specified in the request. The URL will have a `code` parameter that you will use to obtain an access token.

```
https://example.com/your-oauth-url?
code=2710ab6786cd8511a65a1e
```

Your application will use the code parameter, along with your Client ID and Client Secret to exchange the

code for an access token and a refresh token.

cURL

```
curl --location --request POST
'https://app.assignr.com/oauth/token' \
--form 'client_id="[your client id here]"'
\
--form 'client_secret="[your client secret
here]"' \
--form 'code="2710ab6786cd8511a65a1e"' \
--form 'grant_type="authorization_code"' \
--form
'redirect_uri="https://example.com/your-
oauth-url"'
```

Assignr will respond with an access token, and a refresh token. The access token is short-lived (typically two hours).

JSON

```
{
    "access_token":
"678adfb0240e8996534c45ee57c2447a6e7f0ff51a1
    "token_type": "Bearer",
    "expires_in": 7200,
    "refresh_token":
"38437351fc6a4bdd224ea2c83f58c0f42ff18e4faea
    "scope": "read write",
    "created_at": 1606327997
}
```

The access token can now be used in an Authorization header to make requests to the Assignr API on behalf of the user.

cURL

```
curl --location --request GET
'https://api.assignr.com/api/v2/sites/7/game
\
--header 'Accept:
```

```
application/vnd.assignr.v2.hal+json' \
--header 'Content-Type:
application/vnd.assignr.v2.hal+json' \
--header 'Authorization: Bearer
678adfb0240e8996534c45ee57c2447a6e7f0ff51a1c
```

Exchange a refresh token for an access token

Your application should store the refresh token, as this token can be used to obtain a new access token.

cURL

```
curl --location --request POST
'https://app.assignr.com/oauth/token' \
--form 'client_id="[your client id here]"'
\
--form 'client_secret="[your client secret
here]"' \
--form
'refresh_token="38437351fc6a4bdd224ea2c83f58
\
--form 'grant_type="refresh_token"'
```

Assignr will then provide a new access token and refresh token.

JSON

```
{
    "access_token":
"24de2101f51e13aa5dd175d4bb28b64219dabfebc85
    "token_type": "Bearer",
    "expires_in": 7200,
    "refresh_token":
"6b0619df1e97e82f4cf75bf1e71f99bf505369e557f
    "scope": "read write",
    "created_at": 1606328815
}
```

PKCE Support

Our OAuth implementation supports the PKCE protocol (Proof Key for Code Exchange). This process should be used when the application requesting the data is publicly available, like an app on a mobile phone.

Prior to requesting access from the user, the app should generate and temporarily store a random nonce string, 43-128 characters, the `code_verifier` :

Ruby PHP

```
irb(main):004:0> code_verifier =
SecureRandom.hex(32)
=>
"624f67cb8cc1d7ca94748dc9cea681d64e2bce593cc
```

Generate the `code_challenge` hash (SHA256 hash value of the code_verifier, url safe base64 encoded (without trailing "=", example here) :

Ruby PHP

```
irb(main):006:0>
Base64.urlsafe_encode64(Digest::SHA256.diges
=>
"lCRAy4ktIw_Y5Zg0qgm0jR618BSsn9vCwkl90WPAvbE
```

The user should be directed to the authorization URL. The URL is constructed with `client_id` , `redirect_uri` , `response_type` , `scope` , and `code_challenge` and `code_challenge_method` :

Text

```
https://app.assignr.com/oauth/authorize?
client_id=[client_id] \
  &redirect_uri=[encoded-redirect-uri] \
```

```
  &response_type=code \
  &scope=read+write \

&code_challenge=lCRAy4ktIw_Y5Zg0qgm0jR618BSs
\
  &code_challenge_method=S256
```

After authorization, a code is provided as usual. When exchanging the code for access & refresh tokens, add `code_verifier` (the original nonce prior to being hashed), and the `code_challenge_method`.

cURL

```
curl --location --request POST
'https://app.assignr.com/oauth/token' \
--form 'client_id="[your client id here]"'
\
--form 'client_secret="[your client secret
here]"' \
--form 'code="2710ab6786cd8511a65a1e"' \
--form 'grant_type="authorization_code"' \
--form
'redirect_uri="https://example.com/your-
oauth-url"'
--form
'code_verifier="624f67cb8cc1d7ca94748dc9cea6
--form 'code_challenge_method="S256"'
```

Assignr responds with an access token and a refresh token:

JSON

```
{
    "access_token":
"24de2101f51e13aa5dd175d4bb28b64219dabfebc85
    "token_type": "Bearer",
    "expires_in": 7200,
    "refresh_token":
"6b0619df1e97e82f4cf75bf1e71f99bf505369e557f
    "scope": "read write",
    "created_at": 1606328815
```

```
}
```

Updated almost 2 years ago

← Access to the API

Requests & Response Codes →

TABLE OF CONTENTS