Authenticating to REDACTED with OAuth

All our API endpoints are designed to work only for authenticated users, and we identify the user based on the access token passed with any request.

To generate a user access token, you must first request access through a flow based on the OAuth 2.0 authorization framework.[1] Once you have a valid token, include the header `Authorization: Bearer [token]` in each request.

Start the OAuth flow

To start the OAuth flow and request user access, send a user to our OAuth page (at `https://www.redacted.com/oauth/`) with the following parameters:

• **`client_id:`**	This is the unique ID for your app also referred to as App ID.
• **`redirect_uri:`**	This must be one of the redirect URIs registered on your app. The value of this parameter must exactly match the registered value.
• **`response_type:`**	Set this value to the literal string code as specified in the RFC.
• **`scope:`**	This is a comma delimited list of scope names covering the specific data and/or functions you want to access for this user. You must request access to at least one scope.
• **`state`** (optional):	This parameter can be any string value defined by you. This parameter is commonly used to prevent cross-site request forgery. This value is appended to your redirect URI, as part of the request query string, during the OAuth transaction. More information on preventing cross-site request forgery is available from OWASP.

Here is an example request to the OAuth page:

```
https://www.redacted.com/oauth/ client_id={YOUR_CLIENT_ID}&
redirect_uri={YOUR_REDIRECT_URI}& response_type=code &
scope=boards:read,pins:read & state={YOUR_OPTIONAL_STRING}
```

If the user is logged in when clicking on the OAuth link, they will see a page where they can approve access for your app for the scopes you're requesting. If the user is not logged in, they will be asked to log in.

1 This framework is published by the Internet Engineering Task Force and described at https://datatracker.ietf.org/doc/html/rfc6749.

Receive the access code with your redirect URI

Once the user authorizes your app, they will be sent to your redirect URI. We will add the following parameters as we make the call to your redirect URI:

- **`code:`** This is the code you will use in the next step to exchange for an access token.
- **`State:`** This is the optional parameter to prevent cross-site request forgery. Check to make sure it matches what was passed in the first step of the flow. If it does not, the exchange may have been subject to a cross-site request forgery attack.

A redirect URI such as: `https://www.example.com/oauth/`**`redacted`**`/oauth_response/`
will result in a callback request like:
`https://www.example.com/oauth/`**`redacted`**`/oauth_response/?code={CODE}&state={YOUR_OPTIONAL_STRING}`

Exchange the code for an access token

(OAuth 2.0: Access Token Request/Response)

Once you have received the code to your redirect URI, you can exchange it for an access token by making a POST request to our access token endpoint with a request body and content type of:

```
application/x-www-form-urlencoded:
https://api.redacted.com/v5/oauth/token
```

You must use HTTP Basic Authorization to properly authenticate your requests to this endpoint:

- Where username is expected, use `client_id`
- Where password is expected, use `client_secret` (aka `App Secret` or `App Secret Key`).

This means that your request would have a header of:

```
Authorization: Basic {base64 encoded string made of client_id:client_secret}
```

For example:

- If your `client_id` is `123` and your `client_secret` is `456`, you would create a base64 encoding of `123:456`.
- A base64 encoding of `123:456` is `MTIzOjQ1Ng==`
- Therefore, your request header would be `Authorization: Basic MTIzOjQ1Ng==`.

You can use the following command in Linux/OSX systems to quickly get the base64 encoded string (replace with your unique `client_id` and `client_secret`, or store them as environment variables):

```
echo -n $client_id:$client_secret | base64
```

Required Request Body Parameters

- **`Code:`** This is the authorization code received from the previous request.
- **`redirect_uri:`** This is the redirect URI value supplied by you when you registered your app. The value of this parameter must exactly match the registered value.
- **`grant_type:`** Set this value to the literal string `authorization_code` as specified in the RFC.

Here's an example request to exchange a code for an access token:

```
curl -X POST https://api.redacted.com/v5/oauth/token
--header 'Authorization: Basic {base64 encoded string made of
client_id:client_secret}'
--header 'Content-Type: application/x-www-form-urlencoded'
--data-urlencode 'grant_type=authorization_code'
--data-urlencode 'code={YOUR_CODE}'
--data-urlencode 'redirect_uri=http://localhost/'
```

If successful, the response you receive will contain an access token and refresh token with expiration times and relevant scope information. Here's an example:

```
{
  "access_token": "{an access token string prefixed with 'foo-a'}",
  "refresh_token": "{a refresh token string prefixed with 'foo-r'}",
  "response_type": "authorization_code",
  "token_type": "bearer",
  "expires_in": 2592000,
  "refresh_token_expires_in": 31536000,
  "scope": "boards:read boards:write pins:read"
}
```

The values given for `expires_in` and `refresh_token_expires_in` are the lifetime in seconds for your access token and refresh token respectively. The token you receive should contain only the scopes requested during the Start the OAuth flow step. Attempting to use an access token after its lifetime has expired will result in an HTTP 401 status with a specific API error code of `2`.

You may generate a new access token at any point during the lifetime of the refresh token as described in Refresh an Access Token below. Once the refresh token lifetime has expired you will need to explicitly request access again by starting the OAuth flow from the beginning.

Refresh an Access Token

(OAuth 2.0 link: Refresh an Access Token)

Access tokens expire within 30 days (2592000 seconds) of being issued. Refresh tokens last for 365 days (31536000 seconds).

Get a new access token by calling Link Redacted, using HTTP Basic Authorization.

Where username is expected, use `client_id`; where password is expected, use `client_secret` (aka `App Secret` or App Secret Key).

Your request would then have a header of `'Authorization: Basic {base64 encoded string made of client_id:client_secret}'`.

Required Request Body Parameters:

- **`refresh_token:`** This is the full refresh token string (starting with "foo-r") you have already received when exchanging an authorization code for an access token.
- **`grant_type:`** Set this value to the literal string refresh_token as specified in the RFC.

Optional Request Body Parameter:

- **`scope:`** A .csv listing scopes.
  Use this parameter if you'd like to receive an access token that has only a subset of the scopes originally granted.

Here's an example request for refreshing an access token:

```
curl -X POST https://api.redacted.com/v5/oauth/token
--header 'Authorization: Basic {base64 encoded string made of
client_id:client_secret}'
--header 'Content-Type: application/x-www-form-urlencoded'
--data-urlencode 'grant_type=refresh_token'
--data-urlencode 'refresh_token={YOUR_REFRESH_TOKEN}'
--data-urlencode 'scope=boards:read'
```

If successful, the response you receive will contain a new access token with an expiration time and relevant scope information. Here's an example:

```
{
    "access_token": "{an access token string prefixed with 'pina'}",
    "response_type": "refresh_token",
    "token_type": "bearer",
    "expires_in": 2592000,
    "scope": "boards:read"
}
```

Store an access token to a JSON-encoded file *(optional)*

For example, you might store your token to `common/oauth_tokens/access_token.json`

To generate this file, use a language-specific get_access_token script. See example in the API Quickstart on GitHub.

- A complete set of JSON object keys will look like this:

```
{
      "access_token": "string",
      "name": "string",
      "refresh_token": "string"
}
```

- **`access_token`** is the access token returned by the OAuth flow (required)
- **`name`** is a textual description of the access token (e.g. "API test account #2")
- **`refresh_token`** is the refresh token returned by the OAuth flow

Use an access token

Include the access token in the request header as part of the key-value pair for authorization. For the key `Authorization` , use the value `Bearer <your token>`.

For example, in curl, the header for a `GET user_account` request where the token was `123` would look like this: `curl https://api.redacted.com/v5/user_account --header 'Authorization: Bearer 123'`

Additional Resources

- Examples of an OAuth implementation in different programming languages in the API QuickStart on Github (link redacted).
- The Internet Engineering Task Force’s OAuth 2.0 Authorization Framework *(at https://datatracker.ietf.org/doc/html/rfc6749/)*
- OAuth.net's Getting Started
- OAuth.com on OAuth Redirects
- OAuth.com on validating redirect URIs