API Authentication Guide

Authenticate with NationBuilder API using OAuth 2.0

Updated over 11 months ago

Table of contents

Table of Contents

Getting API Access

If you already have an account, just login to the control panel and continue below. To get an account, please apply to become a Certified Developer. You will be given a sandbox account.

Developer tools are only available to NationBuilder certified developers or nations on an Enterprise or Network plan. If you are looking to connect an application that is not already one of our integrations, you can reach out to one of our certified developers or apply to be certified.

Setting Up OAuth 2.0 Authentication

You can obtain access tokens to retrieve data from the NationBuilder API in your application with the OAuth 2.0 authorization code flow. This guide assumes some familiarity with the OAuth 2.0 process. If you'd like to learn more about OAuth, this is a helpful resource.

1. **Register the application**
   Log in to your nation, navigate to Settings > Developer > Register New App. Register your application with a name and OAuth callback URL. The OAuth callback URL is described in more detail below.
   After you register, you'll use the Client ID and Client Secret provided in your OAuth requests. Do not share your Client ID and Client Secret.

2. **Ask a nation's administrator for access**

   First, your application must make a request to NationBuilder to receive a response with a short-lived `code` that can be exchanged for an access token:

   ```
   https://{slug}.nationbuilder.com/oauth/authorize?response_type=code&client_id=...&redi
   ```

   The `{slug}` is the nation slug of the nation from which your application is requesting approval to retrieve data via the NationBuilder API. For example, your application's user could provide this slug via a text field in your application.

   The `client_id` and `redirect_uri` (OAuth callback) are available on your application's page in Settings > Developer > Your apps. The OAuth callback is an endpoint of your system to which NationBuilder's system will return a `code` upon successful authorization.
   Before your application receives a `code` back, your user will be prompted to log in to their NationBuilder account if they aren't already.
   In addition, if your application hasn't been granted access to a nation by this NationBuilder admin before, they will be redirected to this screen to do so:

   This authorization process will occur for every admin that will receive NationBuilder access tokens to use with the client application.

3. **Receive a code**

   After verifying the user is logged in and your application has been approved to access the nation, NationBuilder will complete the `/oauth/authorize` request by redirecting to the OAuth callback you provided when you registered the app with a `code` parameter in the query string:

```
GET http://www.yourapp.com/oauth_callback?code=...
```

Use this code for the next step. This code expires after 10 minutes and does not need to be stored.

Unhappy path: If the admin user denies your application's access, your callback URL will receive the `error=access_denied` param.

4. **Exchange the code for an access token**
   Exchange the `code` returned in the last step for an access token by issuing another request:

   ```
   POST https://{slug}.nationbuilder.com/oauth/token grant_type=authorization_code
   client_id=... client_secret=...
   redirect_uri=... code=...
   ```

   The `client_id`, `client_secret`, and `redirect_uri` (OAuth callback) are available on your application's page in Settings > Developer > Your apps.

   A practical example is using the command line utility cURL like this:

   ```
   curl -X POST --header "Content-Type: application/json" --header "Accept: application/j
   ```

   This request will receive a response like this:

   ```
   { "access_token": "<access_token_here>", "token_type": "bearer", "scope":"default",
   "created_at": 1632773996 }
   ```

   Record the access token from the response you receive to this request.

5. **Use the access token to get data**
   With this access token you can make requests on the user's behalf. See our API endpoint documentation for full details. As an example, this is the request you would use to get the first page of people in a nation:

```
GET https://{slug}.nationbuilder.com/api/v1/people?access_token=...
```

6. **IMPORTANT:** If you have migrated to NationBuilder V2 access tokens, the access tokens you receive back from the OAuth 2.0 process will expire after 24 hours. You also must implement a refresh flow. See the next section.

   The API uses JavaScript Object Notation (JSON). If you receive a 406 response code, it means that you need to include the Content-Type and Accept headers of your request to "application/json".

Refresh Token Flow

At this time, all current and new applications are using NationBuilder's V1 access tokens. You do not need to implement a refresh token flow. However, we recommend migrating to V2 access tokens and implementing one.
If you have migrated your application to use V2 access tokens, they expire after 24 hours. You must implement a refresh token flow as part of your OAuth process to exchange a refresh token for a new access token once it expires.

1. **Store refresh token**
   With V2 tokens, you'll receive a refresh token as well as an access token in the response from the `/oauth/token` request (see *Exchange the code for an access token* above). Your application needs to store this refresh token to use later. You can only use the refresh token to refresh the access token from this response, so your application must associate them.

   ```
   { "access_token": "<access_token_here>", "refresh_token": "<refresh_token_here>", "tok
   "expires_in": 86400, "scope": "default", "created_at": 1632773996 }
   ```

2. **Handle expired tokens**
   Your application must make a `refresh_token` grant type request to receive a new access token before or after the access token expires:

```
POST https://{slug}.nationbuilder.com/oauth/token
     grant_type=refresh_token
     refresh_token=...
```

```
client_id=...
client_secret=...
```

You will receive the same response from a refresh token flow as the original access token exchange, but with a new access token and refresh token value:

```
{
  "access_token": "<access_token_here>",
  "refresh_token": "<refresh_token_here>",
  "token_type": "Bearer",
  "expires_in": 86400,
  "scope": "default",
  "created_at": 1632773996
}
```

Your application can make the refresh flow request either by:

a. Handling the token_expired error response that will result when you make an API request with an already-expired access token:

   ```
   { "statusCode": 401, "data": { "code": "token_expired", "message": "Your access tc
   ```

   Upon receiving this response, it means that the access token you've stored has expired, so you can make the `refresh_token` request above to refresh the access token and receive a new access and refresh token.

b. Refreshing the access token in your application before it expires. When you receive an access token, the response body includes an `expires_in` field with the number of seconds until the access token will expire (24 hours by default). You can use this information to calculate a time within your application to refresh the token before it expires instead of handling the error.

3. **Store new access token and refresh token**
   If this request succeeds because the refresh token is valid to exchange for a new access token, you will receive a new `access_token` and a new `refresh_token` in the response.

   If you refreshed by handling the token_expired error, your application can now use the new access token to retry the API request that originally failed.

The refresh token you used to get a new access token will be revoked upon use. You will need to store the new access and refresh tokens to use again, and can get rid of the old ones if you wish.

Refresh tokens do not expire.

You can test your refresh flow via making a request to refresh an existing V2 access token. Even if your token has not yet expired, a successful request will return a new access and refresh token.

PKCE

PKCE is an additional security layer to the OAuth 2.0 protocol that prevents malicious actors from intercepting the `code` returned to the client and using it to exchange for an access token. We support the optional use of PKCE in our OAuth 2.0 flow. If you include PKCE params with your requests to `/oauth/authorize` and `/oauth/token`, the code challenge will be stored and the code verifier validated. More information about how to implement PKCE can be found here.

Test Tokens

If you need a faster way to begin testing the API, you can use test access tokens. In your nation's control panel, go to Settings > Developer > API token. **Do not use this test token in your production app. Instead, implement the OAuth flow as described above.**

Related Articles

- API Authentication Quick Start guide
- Generating API Tokens
- Access Token Migration Guide
- NationBuilder API QuickStart Guide
- Connecting Zapier to NationBuilder

Did this answer your question?

😞 😐 😃