OAuth 2

DEPRECATED

As part of integrating Manifold's embedded marketplace into your platform, this guide shows you how to authorize Manifold's authentication component to communicate with the Marketplace API on behalf of your users. Manifold's authorization model requires:

- enabling transparent authorization
- implementing a standard OAuth 2.0 flow
- exposing the UserInfo endpoint

See also Authentication Overview to learn how this process works.

Enable Transparent Authorization

In the more familiar OAuth implementation, end users are presented with a screen prompting them to explicitly authorize access to an application.

The Manifold model, however, requires transparent authorization, which means that you must automatically accept Manifold authorization requests on behalf of your users without showing them any prompts. Manifold's authentication and authorization with your environment happen in the background, through an invisible iframe.

Note: Because transparent authorization still performs the authorization code exchange step, it also differs from the OAuth Implicit Grant type.

Provide Scope Information to Manifold

An OAuth scope is a mechanism to limit an application's access to a user's account. The access token issued to the application is then limited to the scopes granted.

Manifold requires access to the following end-user resources:

- user ID
- user email
- user name

If you restrict access to any of these resources through OAuth scopes, you must provide this information to Manifold.

Implement a Standard OAuth 2.0 Flow

This section provides information to guide you in implementing OAuth 2.0 for Manifold authorization. Complete the following tasks:

- Create a Client ID and Client Secret
- Allow OAuth Grants
- Implement the Authorization Endpoint
- Implement the Token Endpoint
- Allow Refresh Tokens

Create a Client ID and Client Secret

As a third-party application requiring access to your platform, Manifold must be registered to use your OAuth server. During this registration process, you assign Manifold:

- a client ID — a public identifier for the Manifold application. This ID is public, but it shouldn't be guessable, so often it takes the form of a 32-character hex string. It must be unique across all clients handled by your authorization server. For examples, see OAuth.com's guide.
- a client secret — a string known only to the Manifold application and your authorization server. To generate a secure secret, you can use a cryptographically secure library to generate a 256-bit value and then convert it into a hex representation. To learn more, see OAuth.com's guide.

Manifold uses the client secret when exchanging a temporary authorization code for an access token.

Note: If you need to manually send the client ID and client secret to us, we can provide you with a PGP public key to sign your client secret to ensure secure transfer.

Allow OAuth Grants

The OAuth 2.0 specification describes a number of “grants”, or methods, for a client application to acquire an access token. The access token is then used to authenticate a request to an API endpoint.

To authorize the end user to access the Manifold Marketplace, you must configure the following OAuth grants for the Manifold client ID:

- **Authorization Code** - used to exchange an authorization code for an access token
- **Refresh Token** - used to exchange a refresh token for an access token when the access token has expired. A refresh token is required so that our background workers can obtain necessary information, such as authorization patterns.

Implement the Authorization Endpoint

The authorization endpoint is used to fetch a temporary auth token from your environment. The expiry for this token should be very short — between 30 and 60 seconds. The token should be single-use only (an auth token shouldn’t be used twice to be exchanged for an access token).

Manifold provides a redirect URI:

```
https://login.manifold.co/oauth/web/callback
```

Manifold assumes that your environment verifies that the end user is authenticated, and Manifold transparently accepts the predefined scopes on the user’s behalf.

Note: If you implement this endpoint on a URL other than `<baseurl>/oauth/login`, you must give us the fully qualified URL.

Request/response flow for authorization endpoint

REQUEST

```
Manifold -> Platform Redirect:
https://<PLATFORM_DOMAIN>/oauth/login?access_type=online&client_id=<CLIENT_ID>&
```

RESPONSE

```
Platform -> Manifold Redirect:
https://login.manifold.co/signin/oauth/callback?code=<AUTH_CODE>&state=<RANDOM_
```

Implement the Token Endpoint

The token endpoint is used to exchange a temporary authorization code for an access token. This exchange occurs transparently in the background, and happens immediately when the callback URL is reached. Manifold uses the provided client secret to exchange the token.

Note: If you implement this endpoint on a URL other than `<baseurl>/oauth/token`, you must give us the fully qualified URL.

Request/response flow for token endpoint

REQUEST

```
POST /oauth/token HTTP/1.1
Host: https://<PLATFORM_DOMAIN>
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip

client_id=<CLIENT_ID>&client_secret=<CLIENT_SECRET>&code=<AUTH_CODE>&grant_type
```

RESPONSE

```
HTTP/1.1 200 OK
Cache-Control: no-store
Content-Type: application/json;charset=UTF-8

```

```
{
   "access_token": "<ACCESS_TOKEN>",
   "expires_in": 7200,
   "refresh_token": "<REFRESH_TOKEN>",
   "token_type":"Bearer"
}
```

Allow Refresh Tokens

Manifold uses background workers to execute specific operations. Using a Refresh Token ensures that we can refresh an expired access token when necessary. Manifold sends the refresh_token grant to request this.

For example, when an end user SSOs into one of Manifold's providers, the refresh token ensures that the provider can fetch up-to-date user information on behalf of Manifold.

Expose the UserInfo Endpoint

The Manifold client makes requests to the UserInfo endpoint using the obtained access token. The UserInfo endpoint is part of OpenID Connect 1.0, which is a simple identity layer on top of the OAuth 2.0 protocol. You must create a UserInfo endpoint according to the OpenID Connect specification.

Examples:

```
UserInfo Request
  GET /oauth/userinfo HTTP/1.1
  Host: <baseurl>
  Authorization: Bearer <access_token>

UserInfo Successful Response:
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
   "sub": "248289761001",
   "name": "Jane Doe",
   "email": "janedoe@example.com",
  }

```

```
UserInfo Error Response:
  HTTP/1.1 401 Unauthorized
  WWW-Authenticate: error="invalid_token",
  error_description="The Access Token expired"
```

Notes:

- If you implement this endpoint on a URL other than `<baseurl>/oauth/userinfo`, you must give us the fully qualified URL.
- In a future version, we'll require the UserInfo endpoint to include additional claims. These claims will be used to represent access roles for the end user.