Authentication

This document provides details on the necessary steps and information to get started with an API integration to StockX. It includes information on API keys, authentication flow, as well as technical information, and examples.

How OAuth2 Authorization Works

StockX uses the OAuth2 Authorization Code flow to authenticate and authorize user requests. In order for your application to access StockX Public APIs and your data within StockX, your application will need to integrate with the Authorization Code Flow to gain access.

Access

StockX Audience: Used to access StockX API services. To access any StockX API service, such as login or bid, please make sure your audience is set to gateway.stockx.com.
Auth Domain: To access the StockX auth service, the domain for authentication endpoints is https://accounts.stockx.com.
Scope: A valid OpenID Connect (OIDC) scope is used by StockX auth service to give the API User application the correct access level. The scope sent must contain at least opened, required by OIDC to verify the user's identity and required to receive a refresh token; a minimal necessary scope to access StockX is `scope=offline_access openid`

Authenticating at StockX as an API User

Authorize the user

First you must authorize the application and the user via the StockX IDP. This is done by sending the user to the `/authorize` endpoint URL with the proper credentials and grant.
For this first authorize call, the response type will be `code`. State is an opaque value that will be defined by the API User, e.g., `state = abcXYZ9876`.

```
GET https://accounts.stockx.com/authorize?
    response_type=code&
    client_id={APPLICATION_ID}&
    redirect_uri={APPLICATION_REDIRECT_URI}&
    scope=offline_access%20openid&
    audience=gateway.stockx.com&
    state={OPAQUE_STATE_VALUE}
```

Parameter Name	Required Value	Description
response_type	code	Denotes the kind of credential returned (`code` or `token`). For this flow, the value must be `code`.
client_id	Provided on the Keys page	Your application's Client ID, will be generated after application creation.
redirect_uri	Example: https://www.yourdomain.com/callback	The URL to which StockX auth service will redirect the browser after the authorization has been granted by the user. The Authorization Code will be available in the `code` URL parameter. You will need to provide this url in your application.
scope	`offline_access openid` (space as delimiter)	Specifies the scopes for which you want to request authorization, which dictate which claims (or user attributes) you want returned. In StockX is `scope=offline_access openid`

Parameter Name	Required Value	Description
audience	gateway.stockx.com	The unique identifier of the API your web app wants to access. This is provided by StockX.
state	Any string API user chooses Example: abcXYZ9876	An opaque arbitrary alphanumeric string your app adds to the initial request that StockX auth service includes when redirecting back to your application. We recommend API users to use this string to validate the incoming callback request, to prevent cross-site request forgery (CSRF) attacks, which refer to https://owasp.org/www-community/attacks/csrf.

Login and receive authorization code

After successfully authorizing your application, the StockX login page will be shown and you must enter the email and password of the API User account. Once login is completed successfully, the response will be a HTTP 302 redirect with a URL. The authorization code is located at the end of the callback URI.

```
HTTP/1.1 302 Found
Location: undefined?code=AUTHORIZATION_CODE&state=xyzABC123
```

The `code=AUTHORIZATION_CODE` value in the URL, specifically the `AUTHORIZATION_CODE` value, is the value that you need. You will use that code on the next step to get tokens.

Note: This is your application's endpoint that you will need to implement to receive the authorization code.

Exchange authorization code for a set of tokens

Now that you have received an Authorization Code, you must POST a new call to `/authorize` to exchange the code for tokens and add the extracted Authorization Code (code) from the previous step to the request.

Below you will find an example of a call to `/authorize` for token exchanging. Keep in mind that the response will give you a set of tokens and that some of them must be stored on your side securely as they will be needed later on to access StockX APIs.

Request

```
curl --request POST \
 --url 'https://accounts.stockx.com/oauth/token' \
 --header 'content-type: application/x-www-form-urlencoded' \
 --data 'grant_type=authorization_code' \
 --data 'client_id={YOUR_CLIENT_ID}' \
 --data 'client_secret={YOUR_CLIENT_SECRET}' \
 --data 'code={YOUR_AUTHORIZATION_CODE}' \
 --data 'redirect_uri={YOUR_CALLBACK_URI}'
```

Parameter Name	Description
grant_type	Set this to `authorization_code`
client_id	Generated at application creation
client_secret	Generated at application creation
code	The `authorization_code` retrieved in step 2
redirect_uri	The URL to which StockX auth service will redirect the browser after authorization has been granted by the user. The `authorization_code` will be available in the code URL parameter.

Response

If you are able to successfully exchange the code for tokens, you will receive a response with a payload containing `access_token`, `refresh_token`, `id_token` , and `token_type`. Please see below an example payload response.

```
{
  "access_token": "eyJz93a...k4laUWw",
  "refresh_token": "GEbRxBN...edjnXbL",
  "id_token": "eyJ0XAi...4faeEoQ",
```

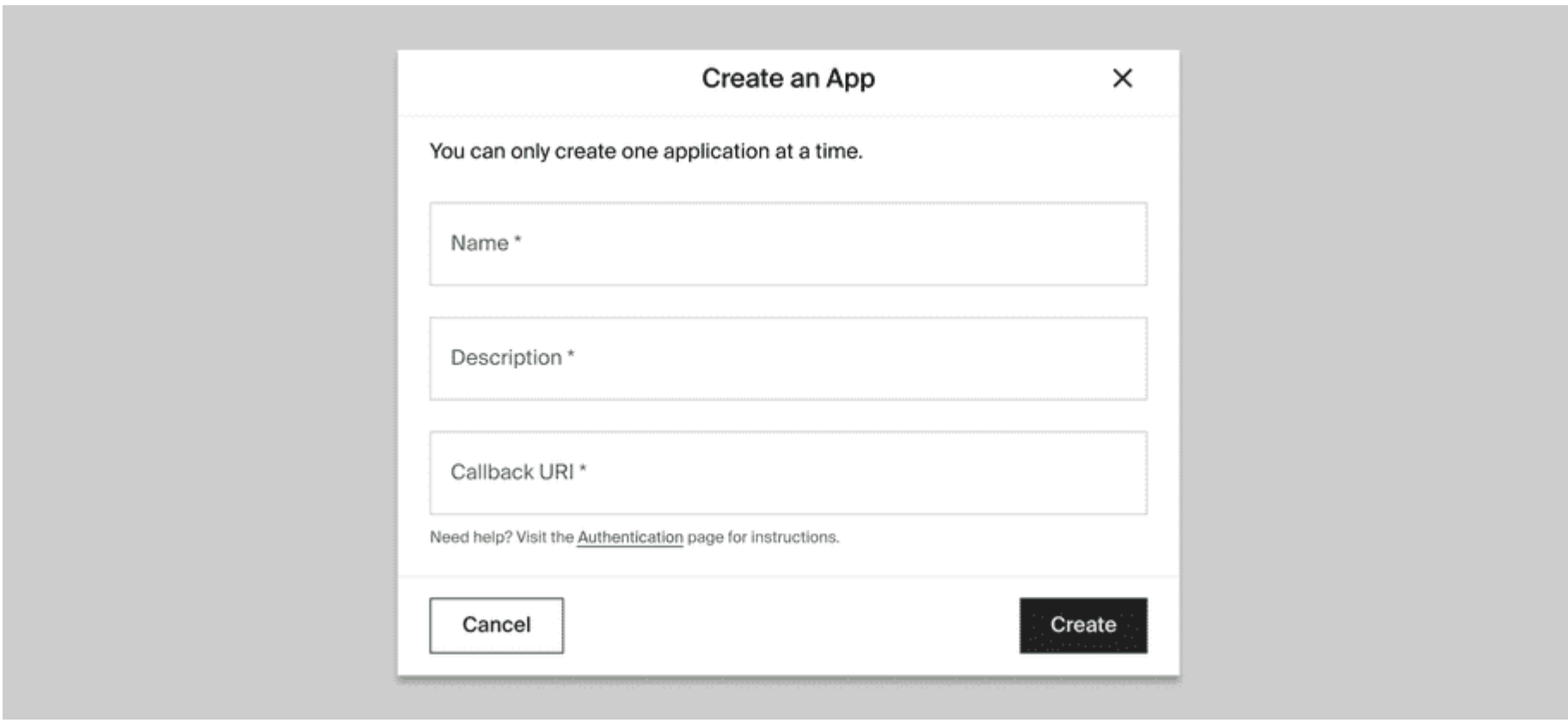
```
  "token_type": "Bearer"
}
```

Application Settings

The following guide shows how to create, update and delete a new application. The app provides the Client ID and Client Secret needed to implement authorization flows.

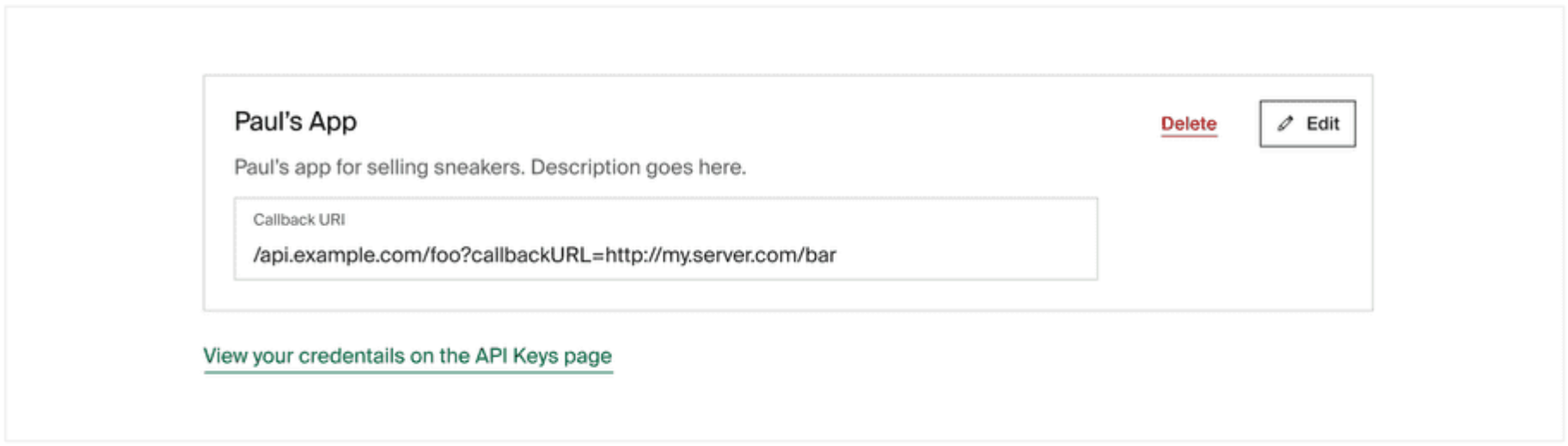
> Note: At this time StockX supports one Application per StockX account

To do so, go to your Applications and click on the Create an App button to open the following dialog box:



Enter your application's name, description, and callback URI of your choosing, and click CREATE. Your application is now created and registered, and you'll see your newly created application on the Applications page.
The callback URI enables the StockX authentication service to automatically invoke your application every time the user logs in.

Application Page



The Applications page provides the following information:

- Application name
- Application description
- Callback URI
- A link to view your application credentials
  - Client ID, the unique identifier of your application
  - Client Secret, the key you will use to authorize your Web API calls

You can update your application by clicking the EDIT button, and you can delete the application by clicking the red DELETE button.

Call StockX API

Once you have obtained an `access_token` from the previous steps, you can start making API calls. The access token must be added to your header request as a bearer token, i.e., with the text `Bearer` in front.
The Production API URL: api.stockx.com
Authorization: Bearer {ACCESS_TOKEN}

x-api-key: {api-key}

A CURL example is shown below of a header with a bearer token

```
curl --location --request GET
'https://{baseUrl}/{version}/{endpoint} \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer {ACCESS_TOKEN}' \
--header 'x-api-key: {api-key}'
```

Header Name	Description
baseUrl	StockX API URL, e.g. api.stockx.com
version	StockX API version, e.g. v1
authorization	Set this to `Bearer `access_token`
x-api-key	Auto-generated after developer access approval, located in the keys page

Note: Please note that the `access_token` has an expiration of 12 hours after which you need to request a new `access_token`, which you can do with the `refresh_token` step, outlined below.

Refresh Token

As a StockX API User once a call is made to the StockX audience with a bearer token, that token is verified. If the `access_token` is no longer valid, it will return you an HTTP 401 and you will need to renew your `access_token`.
To renew an `access_token` you need a `refresh_token`. `refresh_token` is a special token used by the IDP (Identity Provider) to allow you to request a new `access_token` when the one you received during authentication is no longer valid. A `refresh_token` is returned after a login and has a long lifespan, and it will be valid to exchange tokens for as long as your session is valid, so when you call to exchange tokens you will receive a new `access_token` but not a new `refresh_token`, as those are long-lived with your session.
In order to get a refresh token you must call `/oauth/token` with `grant_type=refresh_token`. To exchange a refresh token for an access token you must make the same call. The calls can only be made after you already made the `/authorize` call. A sample request and response to exchange a `refresh_token` or a new `access_token` is shown below:

```
curl --location --request POST \
 --url 'https://accounts.stockx.com/oauth/token' \
 --header 'Content-Type: application/x-www-form-urlencoded' \
 --data-urlencode 'grant_type=refresh_token' \
 --data-urlencode 'client_id={YOUR_CLIENT_ID}' \
 --data-urlencode 'client_secret={YOUR_CLIENT_SECRET}' \
 --data-urlencode 'audience=gateway.stockx.com' \
 --data-urlencode 'refresh_token={YOUR_REFRESH_TOKEN}'
```

Parameter Name	Description
grant_type	Set this to `refresh_token`
client_id	Generated at application creation
client_secret	Generated at application creation
audience	Set this to `gateway.stockx.com`
refresh_token	The Refresh Token

```
{
  "access_token": "ey....",
  "expires_in": 43200,
  "id_token": "ey....",
  "scope": "openid email offline_access",
  "token_type": "Bearer"
}
```

Auth0 SDKs and Documentation

Please review the documentation below for our identity provider - Auth0.
Auth0 Authentication flow:

- https://auth0.com/docs/flows/authorization-code-flow
- https://auth0.com/docs/flows/call-your-api-using-the-authorization-code-flow
- https://auth0.com/docs/api/authentication?http#introduction

Auth0 PHP SDK and Documentation:

- SDK: https://github.com/auth0/auth0-php
- https://auth0.com/docs/quickstart/webapp/php/01-login
- https://auth0.com/docs/libraries/auth0-php
- https://github.com/auth0/auth0-PHP

Auth0 Python Documentation:

- SDK: https://github.com/auth0/auth0-python
- https://github.com/auth0/auth0-python
- https://auth0.com/docs/quickstart/webapp/python/01-login
- https://www.diycode.cc/projects/auth0/auth0-python

Auth0 node documentation:

- SDK: https://auth0.github.io/node-auth0/
- https://auth0.com/docs/quickstart/backend/nodejs/01-authorization
- https://auth0.com/docs/architecture-scenarios/server-api/api-implementation-nodejs
- https://auth0.com/docs/quickstart/webapp/nodejs/01-login#configure-node-js-to-use-auth0

Example Code Snippet

The following code snippet simulates logging into StockX via Auth0, receiving an authorization code, and exchanging that code for an access token that can then be used in follow-up API calls. This code snippet is an example and is not meant for production.

```
"use strict";

/*
 * Do not commit your client ID, secret, or domain to your codebase. These
 * are here for the sake of convenience and should never be in your
 * version control system.
 * */

const AUTH0_CLIENT_ID = "";
const AUTH0_DOMAIN = "";
const AUTH0_CLIENT_SECRET = "";

/*
 * This callback URL is where you will be redirected to after you have
 * logged in. This is the URL you provided to StockX.
 * */

const CALLBACK_URL = "";

/* This redirect URI is where you will be redirected to after the
 * authorization code exchange for the access token.
 * */

const REDIRECT_URI = "";

/*
```

```
 * The unique identifier of the API your web app wants to access. This is provided by StockX.
*/
const AUTH0_AUDIENCE = "urn:gateway.stockx.com";

const GRANT_TYPE = "authorization_code";
const ACCESS_TOKEN_URL = `https://${AUTH0_DOMAIN}/oauth/token`;

const PORT = 8000;

const Auth0Strategy = require("passport-auth0");
const axios = require("axios");
const express = require("express");
const expressSession = require("express-session");
const passport = require("passport");

const session = {
    /*
     * This secret isn’t used by Auth0, but rather by the Express session.
     * It signs the session cookie ID.
     * */
    secret: "SESSION COOKIE ID SIGNER: ADD YOUR OWN",
    cookie: {},
    resave: false,
    saveUninitialized: false,
};

const app = express();

app.use(expressSession(session));

const strategy = new Auth0Strategy(
    {
        domain: AUTH0_DOMAIN,
        clientID: AUTH0_CLIENT_ID,
        clientSecret: AUTH0_CLIENT_SECRET,
        callbackURL: CALLBACK_URL,
    },
    (_, __, ___, profile, done) => done(null, profile)
);

passport.use(strategy);
app.use(passport.initialize());
app.use(passport.session());

const router = express.Router();
const authMiddleware = passport.authenticate("auth0", { scope: "openid email profile", audience: AUTH0_AUD

router.get("/login", authMiddleware, (_, res) => res.redirect("/"));

/*
 * This is where the magic happens. When Auth0 redirects you to the
 * callback, it does so with an authorization code. That authorization code
 * can be exchanged for an access token. The access token is what you will
 * use to gain access to StockX's public API.
 * */

router.get("/callback", async (req, res) => {
    try {
        const { data } = await axios.post(ACCESS_TOKEN_URL, {
            grant_type: GRANT_TYPE,
            client_id: AUTH0_CLIENT_ID,
            client_secret: AUTH0_CLIENT_SECRET,
            redirect_uri: REDIRECT_URI,
```

```
            code: req.query.code,
        });

        return res.status(200).json(data);
    } catch (err) {
        return res.send(err.toString()).status(500);
    }
});

router.get("/", (_, res) =>
    res.send("Sample homepage. To login, navigate your browser to localhost:8000/login.").status(200)
);

app.use("/", router);

app.listen(PORT, () => {
    console.log(`Listening to requests on http://localhost:${PORT}`);
});
```