

Connecting with Python

This guide explains how to establish a connection between a Python Flask application and a Keycloak identity provider using Flask-OIDC. It walks through the necessary setup, configuration, and usage of a protected route that requires authentication.

Variables

Certain parameters must be provided to integrate a Python Flask application with Keycloak. Below is a breakdown of each required variable, its purpose, and where to find it. Here's what each variable represents:

Variable	Description	Purpose
<code>CLIENT_ID</code>	Client ID from the Keycloak Clients page	Identifies the Flask app in the Keycloak realm
<code>CLIENT_SECRET</code>	Secret from the Credentials tab of the client	Authenticates the Flask app with Keycloak
<code>ISSUER</code>	Full Keycloak realm URL (e.g. <code>https://your-domain/realms/your-realm</code>)	Defines the OpenID Connect issuer
<code>REDIRECT_URI</code>	The callback URL Keycloak will redirect to after login	Used by Flask-OIDC to complete login flow
<code>TOKEN_ENDPOINT</code>	Token URL from Keycloak	Used for exchanging authorization codes for access tokens
<code>USERINFO_ENDPOINT</code>	User info endpoint from Keycloak	Used to fetch user profile after login

These values can be found in the **Keycloak Admin Console** under **Clients** → **[Your Client]** → **Settings / Credentials / Endpoints**. Make sure to copy and add them to the code as shown.

Prerequisites

Install Python and pip

Check if Python is installed by running:

```
python3 --version
```

If not installed, download it from <https://python.org> and install.

Verify pip installation:

```
pip3 --version
```

Install Required Packages

Install the required Python packages using:

```
pip3 install flask flask-oidc
```

Code

Once all prerequisites are set up, create a new file named `app.py` and add the following code:

```
from flask import Flask, redirect, url_for, jsonify
from flask_oidc import OpenIDConnect

app = Flask(__name__)

# Keycloak OIDC configuration (no JSON file required)
app.config.update({
    'SECRET_KEY': 'your-random-secret',
    'OIDC_CLIENT_SECRETS': {
        "web": {
            "client_id": "CLIENT_ID",
            "client_secret": "CLIENT_SECRET",
            "auth_uri": "https://your-keycloak-domain/realms/your-realm/protocol/openid-
connect/auth",
            "token_uri": "https://your-keycloak-domain/realms/your-realm/protocol/openid-
connect/token",
            "userinfo_uri": "https://your-keycloak-domain/realms/your-realm/protocol/openid-
connect/userinfo",
            "redirect_uris": ["http://localhost:5000/oidc/callback"]
        }
    },
    'OIDC_SCOPES': ['openid', 'email', 'profile'],
    'OIDC_CALLBACK_ROUTE': '/oidc/callback',
```

```

    'OIDC_COOKIE_SECURE': False
})

oidc = OpenIDConnect(app)

@app.route('/')
def index():
    return 'Welcome to the public route.'

@app.route('/protected')
@oidc.require_login
def protected():
    user_info = oidc.user_getinfo(['email', 'sub', 'name'])
    return jsonify({
        "message": "You are authenticated",
        "user": user_info
    })

@app.route('/logout')
def logout():
    oidc.logout()
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(debug=True)

```

Replace the placeholders in the `client_id`, `client_secret`, and URL fields with actual values from your Keycloak instance.

Execution

Open the terminal and navigate to the directory where `app.py` is saved. Once in the correct directory, run the script with the command:

```
python3 app.py
```

If the connection is successful:

1. Open `http://localhost:5000` in your browser — Public route.
2. Open `http://localhost:5000/protected` — Redirects to Keycloak login.
3. After logging in, you'll see user info returned from the protected route.

4. Visit <http://localhost:5000/logout> to end the session and return to the public page.

Revision #1

Created 2025-06-17 07:47:20 UTC

Updated 2025-06-17 09:03:48 UTC