zendesk developers

API Basics > Authentication

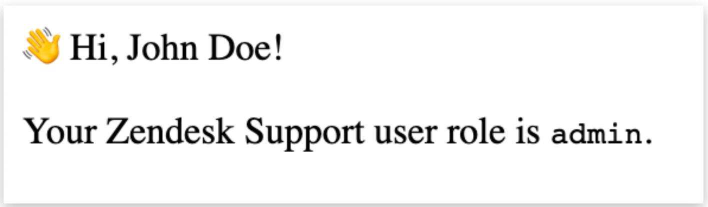
Using OAuth to authenticate Zendesk API requests in a web app

ON THIS PAGE

OAuth 2.0 authentication is a secure way for users to give an app limited access to their Zendesk data without giving away their password. As an app developer, you can use OAuth to safely access Zendesk data on behalf of your users.

In this tutorial, you'll build a basic web app that uses OAuth to authenticate users with Zendesk and retrieve a related OAuth access token. The app then uses this access token to authenticate a call to the Zendesk Support API. The API call retrieves the user's profile data from Zendesk Support.

After making the call, the app displays a webpage that contains the user's Zendesk Support user name and role. You can use the app's code as a starting point for building your own OAuth flows with Zendesk.



👋 Hi, John Doe!

Your Zendesk Support user role is `admin`.

Disclaimer: Zendesk provides this article for instructional purposes only. Zendesk doesn't provide support for the apps or example code in this tutorial. Zendesk doesn't support third-party technologies, such as Node.js, Python, or related libraries.

Note: This article covers how to use OAuth to access Zendesk data in an external app or system. To use OAuth to access data from an external system in a Zendesk app, see Adding third-party OAuth to a Support app.

Authorization code grant type

Zendesk supports the OAuth 2.0 authorization code grant type, also called the authorization code flow. For an overview of the authorization code grant type and its workflow, see the Authorization code grant type in Zendesk help.

Token expiration

Your OAuth client should implement a fallback mechanism to handle expired access tokens and expired refresh tokens. For example, if the access token expired or encounters an error, you can refresh it. However, if the refresh process fails or there is no refresh token linked to the access token, you must redirect the user to `https://{subdomain}.zendesk.com/oauth/authorizations/new` to re-authorize your application. If the user has previously authorized a token with the same scopes for your OAuth app, and that token is still valid and has not been removed from the Zendesk system, they will not need to re-authorize the app. If the token expired and has been removed, or if the app requests different scopes, the user will be prompted to grant access to the OAuth app. Zendesk does not set the access token `expires_in` value. However,if you decide to configure them through the API, those settings are enforced.

What you'll need

To complete this tutorial, you'll need the following:

- A Zendesk Support account with admin access. To get a free account for testing, see Getting a trial or sponsored account for development.
- A text editor
- A command-line terminal, such as Windows Terminal or Terminal on Mac
- One of the following programming languages:
  - Node.js 18.15.0 or later
  - Python 3.11.2 or later

Registering your app with Zendesk Support

Before you can set up OAuth for an app, you need to register the app with Zendesk Support. You do this by creating an OAuth client.

To create an OAuth client, follow the instructions in Registering your application with Zendesk in Zendesk help. You must be signed in as a Zendesk Support admin to create an OAuth client.

When registering the app, specify `http://localhost:3000/zendesk/oauth/callback` as one of the **Redirect URLs**. You'll create this URL later in the tutorial. Provide any **Client name** and **Unique identifier** you want.

After you create the client, securely save the client's **Unique identifier** and **Secret**. These credentials are also called the client id and secret. You'll use these credentials later in the tutorial.

Note: OAuth clients are scoped to one Zendesk instance. To request a global OAuth client that works with multiple Zendesk instances, see Set up a global OAuth client.

Creating a web app that uses OAuth

Next, create a web app that uses OAuth to connect to Zendesk. The app should handle each step of the OAuth process:

- Authenticating the user with Zendesk
- Exchanging an authorization code for an access token
- Making an authenticated Zendesk API call on behalf of the user. This app makes a call to the Show Self endpoint to retrieve the user's name and role.

Instructions for creating the app are provided for both Node.js and Python. Use the programming language you prefer:

- Node.js
- Python

Node.js

Use the following instructions to create the web app using JavaScript for Node.js.

1. In your terminal, create and navigate to a folder named **oauth_app_example**. Example:

```
mkdir oauth_app_example
cd oauth_app_example
```

2. In the folder, use the following command to create an npm package.

```
npm init -y
```

3. Install dependencies for the project. This project requires the express.js and axios libraries.

```
npm install express axios
```

4. In the **oauth_app_example** folder, create an **app.js** file. Paste the following code into the file.

```
const axios = require("axios")
const express = require("express")
const querystring = require("querystring")

const app = express()
const port = 3000

// In production, store your client id and secret in environment variables
const ZENDESK_CLIENT_ID = "YOUR_CLIENT_ID"
const ZENDESK_CLIENT_SECRET = "YOUR_CLIENT_SECRET"
const ZENDESK_SUBDOMAIN = "YOUR_ZENDESK_SUBDOMAIN"
const REDIRECT_URI = `http://localhost:${port}/zendesk/oauth/callback`

app.get("/", (req, res) => {
  res.send('<p><a href="/zendesk/auth">Sign in to Zendesk</a></p>')
})

app.get("/zendesk/auth", (req, res) => {
  res.redirect(
    `https://${ZENDESK_SUBDOMAIN}.zendesk.com/oauth/authorizations/new?${
querystring.stringify(
      {
        response_type: "code",
        redirect_uri: REDIRECT_URI,
        client_id: ZENDESK_CLIENT_ID,
        scope: "users:read"
      }
    )}`
  )
})

app.get("/zendesk/oauth/callback", async (req, res) => {
  const tokenResponse = await axios.post(
    `https://${ZENDESK_SUBDOMAIN}.zendesk.com/oauth/tokens`,
    {
      grant_type: "authorization_code",
      code: req.query.code,
      client_id: ZENDESK_CLIENT_ID,
      client_secret: ZENDESK_CLIENT_SECRET,
      redirect_uri: REDIRECT_URI,
      scope: "users:read"
    },
    { headers: { "Content-Type": "application/json" } }
  )

  // In production, you'd store the access token somewhere in your app,
  // such as a database.
  const access_token = await tokenResponse.data.access_token

  const profileResponse = await axios.get(
    `https://${ZENDESK_SUBDOMAIN}.zendesk.com/api/v2/users/me.json`,
    { headers: { Authorization: `Bearer ${access_token}` } }
  )

  res.send(`
    <p>👋 Hi, ${profileResponse.data.user.name}!</p>

    <p>Your Zendesk Support user role is <code>${profileResponse.data.user.role}</code>.</p>
`)
})

app.listen(port, () => {
  console.log(
    `Server running on port ${port}. Visit http://localhost:${port}`
  )
})
```

In the code, replace the following placeholders:

- "YOUR_CLIENT_ID"
- "YOUR_CLIENT_SECRET"
- "YOUR_ZENDESK_SUBDOMAIN"

5. To start a local web server using the app, run the following terminal command from the **oauth_app_example** folder:

```
node app.js
```

To stop the server, press Ctrl+C.

Python

Use the following instructions to create the web app using Python.

1. In your terminal, create and navigate to a folder named **oauth_app_example**. Example:

```
mkdir oauth_app_example
cd oauth_app_example
```

2. Install dependencies for the project. This project requires the requests and Flask libraries.

```
pip3 install requests flask
```

3. In the **oauth_app_example** folder, create an **app.py** file. Paste the following code into the file.

```
from urllib.parse import urlencode

import requests
from flask import Flask, redirect, request

app = Flask(__name__)
PORT = 3000


# In production, store your client id and secret in environment variables
ZENDESK_CLIENT_ID = "YOUR_CLIENT_ID"
ZENDESK_CLIENT_SECRET = "YOUR_CLIENT_SECRET"
ZENDESK_SUBDOMAIN = "YOUR_ZENDESK_SUBDOMAIN"
REDIRECT_URI = f"http://localhost:{PORT}/zendesk/oauth/callback"


@app.route("/")
def index():
    return "<p><a href='/zendesk/auth'>Sign in to Zendesk</a></p>"


@app.route("/zendesk/auth", methods=["GET"])
def auth():
    parameters = urlencode(
        {
            "response_type": "code",
            "client_id": ZENDESK_CLIENT_ID,
            "redirect_uri": REDIRECT_URI,
            "scope": "users:read",
        }
    )
    return redirect(
        f"https://{ZENDESK_SUBDOMAIN}.zendesk.com/oauth/authorizations/new?{
parameters}"
```

```
    )


@app.route("/zendesk/oauth/callback", methods=["GET"])
def auth_callback():
    token_response = requests.post(
        f"https://{ZENDESK_SUBDOMAIN}.zendesk.com/oauth/tokens",
        data={
            "grant_type": "authorization_code",
            "client_id": ZENDESK_CLIENT_ID,
            "client_secret": ZENDESK_CLIENT_SECRET,
            "redirect_uri": REDIRECT_URI,
            "code": request.args["code"],
            "scope": "users:read",
        },
    )

    # In production, you'd store the access token somewhere in your app,
    # such as a database.
    access_token = token_response.json()["access_token"]

    profile_response = requests.get(
        f"https://{ZENDESK_SUBDOMAIN}.zendesk.com/api/v2/users/me.json",
        headers={"Authorization": f"Bearer {access_token}"},
    )

    user_name = profile_response.json()["user"]["name"]
    user_role = profile_response.json()["user"]["role"]
    return f"""<p>👋 Hi, {user_name}!</p>

          <p>Your Zendesk Support user role is <code>{user_role}</code>.</p>

          """


if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=PORT)
```

In the code, replace the following placeholders:

- "YOUR_CLIENT_ID"
- "YOUR_CLIENT_SECRET"
- "YOUR_ZENDESK_SUBDOMAIN"

4. To start a local web server using the app, run the following terminal command from the **oauth_app_example** folder:

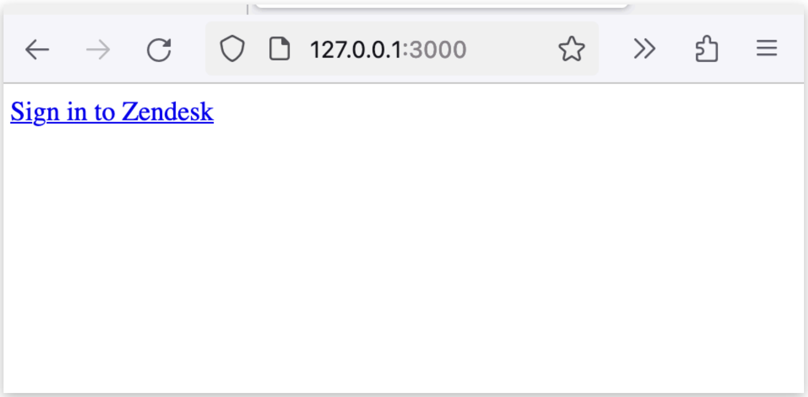
```
python app.py
```

To stop the server, press Ctrl+C.

Testing the app's OAuth flow

To finish, test your app to ensure the OAuth flow works as intended.

1. Use your app to start a local web server.
2. In a web browser, go to `http://localhost:3000` and click **Sign in to Zendesk**.

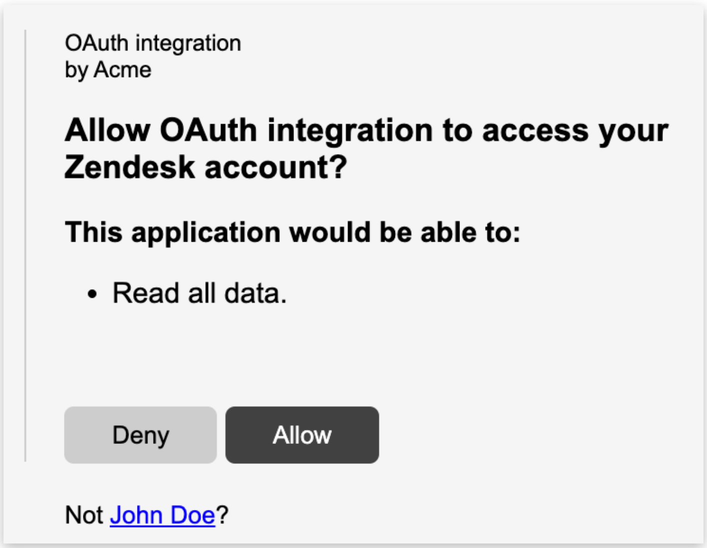


127.0.0.1:3000

Sign in to Zendesk

3. If needed, sign in to Zendesk.

   A Zendesk OAuth page opens in a new tab.



OAuth integration
by Acme

Allow OAuth integration to access your Zendesk account?

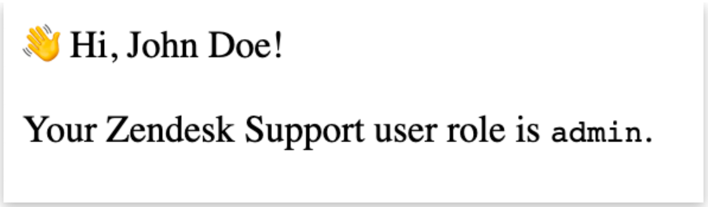
This application would be able to:

- Read all data.

Not John Doe?

4. Click **Allow**.

   The OAuth page closes and redirects back to the app. The app displays a web page that contains your user name and role.


👋 Hi, John Doe!

Your Zendesk Support user role is `admin`.

If you wanted, you can repeat this process using multiple browsers and Zendesk users.

Congratulations! You've created a web app connects with Zendesk using OAuth. As a next step, you can expand the app and get it ready for production. Consider tackling the following tasks:

- Update the app's code to store the client id and secret in environment variables. This helps ensure you don't accidentally share them.
- Add error handling for the app's HTTP requests
- If your app serves users across multiple Zendesk instances, set up a global OAuth client. Then edit the app to include a form that allows users to specify their subdomain
- Update the app to securely store the user's Zendesk access token so you can retrieve it during later sessions. For example, you may store the token in a database used by the app.