Quickstart guide

Introduction

This guide will walk you through retrieving data from providers such as ENS, LENS, and UD using the Domains API.

Prerequisites

- Node.js and npm installed on your machine
- Basic knowledge of JavaScript, React, and Express.js

Step-by-step guide

Step 1: Initialization

1. Create a new directory for your project:

```
mkdir domains && cd domains
```

2. Initialize a new Node.js project:

```
npm init -y
```

3. Install Express, CORS, and Axios:

```
npm install express cors axios
```

4. Install dotenv for securely storing environment variables:

```
npm install dotenv
```

Then create a new file called `.env` and add your DevPortal API key to it:

```
API_KEY=YOUR_1INCH_API_KEY
```

Step 2: Import required libraries

1. Create a new file named `api.js` in your project directory.
2. Add the following code to import necessary packages and initialize your Express application:

```
const express = require("express");
const axios = require("axios");
const dotenv = require("dotenv");
const cors = require("cors");
const path = require("path");

// Load environment variables
dotenv.config({ path: path.resolve(__dirname, ".env") });

const app = express();
app.use(cors());
```

Step 3: Define API endpoints

Add the following code to define your API endpoints:

- Retrieve domain information

```
const BASE_URL = "https://api.1inch.dev/domains/v2.0";

app.get("/api/:domain/info", async (req, res) => {
  const domain = req.params.domain;
  try {
    const constructedUrl = `${BASE_URL}/${domain}/lookup`;
    const response = await axios.get(constructedUrl, {
      headers: {
        Authorization: `Bearer ${process.env.API_KEY}`,
      },
    });
    res.json(response.data);
  } catch (error) {
    res.status(500).json({ error: "API error" });
  }
});
```

- Reverse lookup for a domain:

```
app.get("/api/:domain/reverseinfo", async (req, res) => {
  const domain = req.params.domain;
  try {
    const constructedUrl = `${BASE_URL}/${domain}/reverse-lookup`;
    const response = await axios.get(constructedUrl, {
      headers: {
        Authorization: `Bearer ${process.env.API_KEY}`,
      },
    });
    res.json(response.data);
  } catch (error) {
    res.status(500).json({ error: "API error" });
  }
});
```

- Retrieve provider data with avatars:

```
app.get("/api/:domain/get-providers-data-with-avatar", async (req, res) => {
  const domain = req.params.domain;
  try {
    const constructedUrl = `${BASE_URL}/get-providers-data-with-avatar`;
    const response = await axios.get(constructedUrl, {
      headers: {
        Authorization: `Bearer ${process.env.API_KEY}`,
      },
    });
    res.json(response.data);
  } catch (error) {
    res.status(500).json({ error: "API error" });
  }
});
```

Step 4: Start the server

1. Add the following code to start your Express server and save the file:

```
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

2. Run the server by using in your Terminal:

```
node api.js
```

Response models

The API returns structured JSON responses, which you can use to integrate with your application.

- Response model: Provider data with avatar

```
{
  "result": {
    "protocol": "string",
    "domain":  "string",
    "address":  "string",
    "avatar":  {}
  }
}
```

- Response model: Domain or address information

```
    {
        "result":  {
            "protocol":  "string",
            "address":  "string",
            "checkUrl":  "string"
        }
    }
```

Previous
‹ Introduction

Next
Returns domains for address if existed ›