Ignore sdk yaml file apiData.yaml How to Create SDK Queries in React.

This guide walks you through generating SDK queries from a Swagger/OpenAPI specification in a React project using Vite and @7nohe/openapi-react-query-codegen.

1. Create a React Project with Vite

First, initialize a new React project:

```
npm create vite@latest my-app
cd my-app
npm install
```

2. Add a Swagger/OpenAPI YAML File

Inside your project, create a file named petstore.yaml and place your Swagger/OpenAPI content inside.

How to Generate Swagger YAML from JSON

1. Copy the Swagger/OpenAPI JSON URL from your API provider (for example, https://example.com/swagger/v1/swagger.json).

2. Open Swagger Editor (https://editor.swagger.io/).

3. Paste the JSON into the editor.

4. Export or copy the generated YAML version.

5. Save it as petstore.yaml in your project root (or any preferred location).

3. Install OpenAPI React Query Codegen

Install the package as a development dependency:

```
npm install -D @7nohe/openapi-react-query-codegen
```

4. Configure Code Generation

Add a script to your package.json:

```
{
  "scripts": {
    "codegen": "openapi-rq -i ./petstore.yaml -o ./src/sdk"
  }
}
```

This command:
- Uses petstore.yaml as the input specification.
- Outputs generated SDK queries into ./src/sdk.

5. Run the Code Generator

You can run the codegen script with:

```
npm run codegen
```

Or directly using npx without installing it locally:

```
npx @7nohe/openapi-react-query-codegen openapi-rq -i ./petstore.yaml -o ./src/sdk
```

Install React Query

Install the React Query package:

```
npm install @tanstack/react-query
```

Configure TypeScript

In your tsconfig.app.json file, set verbatimModuleSyntax to false:

```
"verbatimModuleSyntax": false
```

Configure OpenAPI Base URL

In your generated OpenAPI configuration file (e.g., openAPI.ts), set the base URL:

```
export const OpenAPI: OpenAPIConfig = {

  BASE: 'https://fakerestapi.azurewebsites.net',   /////// Add your project url
  CREDENTIALS: 'include',
  ENCODE_PATH: undefined,

};
```

Provide QueryClient in main.tsx

In your main.tsx, wrap the App component with QueryClientProvider:

```
import React from "react";
import ReactDOM from "react-dom/client";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import App from "./App";

const queryClient = new QueryClient();


 <QueryClientProvider client={queryClient}>
     <App />
  </QueryClientProvider>
```

6. Import and Use SDK Queries

After running codegen, you'll have auto-generated SDK files inside src/sdk.
You can now import and use them with React Query in your components.

```
import { useGetPets } from "@/sdk";

function PetList() {
  const { data, isLoading } = useGetPets();

  if (isLoading) return <p>Loading...</p>;

  return (
    <ul>
      {data?.map((pet) => (
        <li key={pet.id}>{pet.name}</li>
      ))}
    </ul>
  );
}

export default PetList;
```

Use version node v20.18.1

```
"devDependencies": {
   "@7nohe/openapi-react-query-codegen": "^1.6.2",
   "@eslint/js": "^9.36.0",
   "@types/node": "^24.6.0",
   "@types/react": "^19.1.16",
   "@types/react-dom": "^19.1.9",
   "@vitejs/plugin-react": "^5.0.4",
   "eslint": "^9.36.0",
   "eslint-plugin-react-hooks": "^5.2.0",
   "eslint-plugin-react-refresh": "^0.4.22",
   "globals": "^16.4.0",
   "typescript": "~5.8.3",
   "typescript-eslint": "^8.39.1",
   "vite": "^7.1.2"
 }
```

```
  "typescript": "~5.8.3",
    "typescript-eslint": "^8.39.1",
    "vite": "^7.1.2"
```

For this error only

```
> test@0.0.0 codegen
> openapi-rq -i ./petstore.yaml -o ./src/sdk

✨ Creating Fetch client
✨ Done! Your client is located in: /home/lnv-20/Pictures/test/src/sdk/requests
file:///home/lnv-20/Pictures/test/node_modules/@7nohe/openapi-react-query-codegen/dist/s
ervice.mjs:31
            throw new Error("Method block not found");
                  ^

Error: Method block not found
    at
file:///home/lnv-20/Pictures/test/node_modules/@7nohe/openapi-react-query-codegen/dist/s
ervice.mjs:31:19
    at Array.map (<anonymous>)
```