Quickstart guide

Gas Price API Tutorial

Here we're going to run through a full stack tutorial on the Gas Price API

As a background here is an short brief into the three parameters we will be getting as a response `baseFee`, `maxPriorityFeePerGas`, and `maxFeePerGas`.

1. Base Fee:
   - Introduced by EIP-1559, the base fee is an amount of ether that is burned (removed from circulation) for every transaction or smart contract execution.
   - The base fee is adjusted by the protocol with every block. Its purpose is to ensure that the Ethereum block (space for transactions) is neither too full nor too empty. If a block is more than 50% full, the base fee increases. If it's less than 50% full, the base fee decreases.
   - The base fee is burned, meaning it's not given to the miners but removed from the total ether supply.
2. Max Priority Fee Per Gas (often called Tip):
   - While the base fee is burned, miners still need an incentive to include your transaction in the block, especially when the network is busy.
   - This is where the "max priority fee" comes in. It's a tip that goes directly to the miner. If you provide a higher tip, miners are more likely to prioritize your transaction.
   - Unlike the base fee, the priority fee isn't algorithmically adjusted. It's set by the sender, and it represents how much they're willing to pay to prioritize their transaction.
3. Max Fee Per Gas:
   - This is the total maximum amount (in wei, the smallest unit of ether) a user is willing to pay per unit of gas.
   - It includes both the base fee and the priority fee (tip).
   - When a transaction is mined, the difference between the max fee per gas and the actual cost (base fee + tip) is refunded to the user.

For more info on these fees and how they're calculated you can read this article here.

To kick off the tutorial we're going to set up the backend with express js for the web server, and axios as an HTTP client to get the information.

Backend

1. Setup

```
mkdir gas-price-app
cd gas-price-app
npm init -y
npm install express axios
```

1. server.js

```
const express = require("express");
const axios = require("axios");
const path = require("path");
const app = express();
const PORT = 3000;

// Serve static files from the React frontend app -> coming in the next part
app.use(express.static(path.join(__dirname, "client/build")));

// Get the front end file by going to the root domain
app.get("/", (req, res) => {
  res.sendFile(path.join(__dirname + "/client/build/index.html"));
});
app.get("/gas-price", async (req, res) => {
  try {
    const response = await axios.get("<https://api.1inch.dev/gas-price/v1.4/1>");
    res.json(response.data);
  } catch (error) {
    res.status(500).json({ message: "Error fetching gas prices" });
  }
});

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

Frontend

For the frontend, we'll use React and we'll also install the HTTP client axios to get the endpoint we just created:

1. Setup

```
npx create-react-app client
cd client
```

```
npm install axios, react
```

2. App.js

Replace the contents of `src/App.js` with:

```
import React, { useState, useEffect } from "react";
import axios from "axios";

function App() {
  const [gasPrice, setGasPrice] = useState({});

  useEffect(() => {
    const fetchGasPrice = async () => {
      const response = await axios.get("/gas-price");
      setGasPrice(response.data);
    };

    fetchGasPrice();
  }, []);

  return (
    <div className="App">
   <h1>Ethereum Gas Prices</h1>
   <div>
    <strong>Base Fee:</strong> {gasPrice.baseFee}
   </div>
   {gasPrice.low && (
        <>
     <div>
      <strong>Low:</strong> {gasPrice.low.maxPriorityFeePerGas}
     </div>
     <div>
      <strong>Medium:</strong> {gasPrice.medium.maxPriorityFeePerGas}
     </div>
     <div>
      <strong>High:</strong> {gasPrice.high.maxPriorityFeePerGas}
     </div>
     <div>
      <strong>Instant:</strong> {gasPrice.instant.maxPriorityFeePerGas}
     </div>
    </>
      )}
  </div>
  );
}

export default App;
```

3. Add proxy to `client/package.json`:

```
"proxy": "<http://localhost:3000>"
```

4. Run both the backend and frontend:

- In the `gas-price-app` directory, run:

```
node server.js
```

- In the `gas-price-app/client` directory, run:

```
npm run build
npm start
```

When you navigate to `http://localhost:3000`, the frontend should now display the gas prices fetched from the 1inch API through our backend server.

And that's it! With just a few lines of code, you've created a full stack application that displays Ethereum gas prices from the 1inch API.

Previous
‹ Introduction

Next
Gas Price from network ›