Quick start

In this guide, we will walk you through the process of performing a token swap using the 1inch Aggregation Protocol. We'll start by checking token allowances, creating the allowance/approval transaction, and finally making the swap. Please note that this example uses the Binance Chain network, but you can replicate it on any network supported by the 1inch Aggregation Protocol.

Let's get started!

How to Use the 1inch Swap API

Step 1: Install the required modules

```
npm install node-fetch@2 web3 yesno
```

💡 Note that if you're installing node-fetch and you can specify the version to be 2.0.0
If you do `node install node-fetch` from npm by default you'll import the version ≥3.3.2 which is an ESM required module requiring you to use ********`import fetch from "node-fetch"`* to reference the module in your .js file
For documentation on the packages please view node-fetch, web3 and yesno
Step 2: Set up your environment

```
const Web3 = require("web3");
const fetch = require("node-fetch");
const yesno = require("yesno");

const chainId = 56; // Chain ID for Binance Smart Chain (BSC)
const web3RpcUrl = "https://bsc-dataseed.binance.org"; // URL for BSC node
const walletAddress = "0x...xxx"; // Your wallet address
const privateKey = "0x...xxx"; // Your wallet's private key. NEVER SHARE THIS WITH ANYONE!
```

Step 3: Define your swap parameters

```
const swapParams = {
  src: "0x111111111117dc0aa78b770fa6a738034120c302", // Token address of 1INCH
  dst: "0x1af3f329e8be154074d8769d1ffa4ee058b1dbc3", // Token address of DAI
  amount: "100000000000000000", // Amount of 1INCH to swap (in wei)
```

```
  from: walletAddress,
  slippage: 1,  // Maximum acceptable slippage percentage for the swap (e.g., 1 for 1%)
  disableEstimate: false,  // Set to true to disable estimation of swap details
  allowPartialFill: false  // Set to true to allow partial filling of the swap order
};
```

Step 4: Define API URLs, Your API Key here and initialize Web3 libraries

```
const broadcastApiUrl = "https://api.1inch.dev/tx-gateway/v1.1/" + chainId + "/broadcast";
const apiBaseUrl = "https://api.1inch.dev/swap/v6.0/" + chainId;
const web3 = new Web3(web3RpcUrl);
const headers = { headers: { Authorization: "Bearer YOUR_API_KEY", accept: "application/json" } };
```

Step 5: Define Helper Functions

```
// Construct full API request URL
function apiRequestUrl(methodName, queryParams) {
  return apiBaseUrl + methodName + "?" + new URLSearchParams(queryParams).toString();
}

// Post raw transaction to the API and return transaction hash
async function broadCastRawTransaction(rawTransaction) {
  return fetch(broadcastApiUrl, {
    method: "post",
    body: JSON.stringify({ rawTransaction }),
    headers: { "Content-Type": "application/json", Authorization: "Bearer YOUR-API-KEY" }
  })
    .then((res) => res.json())
    .then((res) => {
      return res.transactionHash;
    });
}

// Sign and post a transaction, return its hash
async function signAndSendTransaction(transaction) {
  const { rawTransaction } = await web3.eth.accounts.signTransaction(transaction, privateKey);

  return await broadCastRawTransaction(rawTransaction);
}
```

Step 6: Check Token Allowance

```
const allowance = await checkAllowance(swapParams.src, walletAddress);
console.log("Allowance: ", allowance);
```

> **NOTE**
>
> If the output shows `Allowance: 0`, it means that the 1inch router does not have access to swap this token within your wallet.

Creating the Token Allowance (Approval) Transaction

Before the 1inch aggregation protocol can access tokens in your wallet, you need to create an approval transaction. This transaction specifies that the 1inch router is allowed to swap a specific amount of the token you choose.

> ⚠ CAUTION
>
> Please be cautious as approval transactions require payment of a blockchain gas fee, which is deducted in the form of native tokens from your wallet.

Step 1: Set up your environment (same as before)

Step 2: Implement Helper Functions (same as before)

Step 3: Build the Body of the Transaction

```
async function buildTxForApproveTradeWithRouter(tokenAddress, amount) {
  const url = apiRequestUrl("/approve/transaction", amount ? { tokenAddress, amount } : { tokenAddress });

  const transaction = await fetch(url, headers).then((res) => res.json());

  const gasLimit = await web3.eth.estimateGas({
    ...transaction,
    from: walletAddress
  });

  return {
    ...transaction,
    gas: gasLimit
  };
}

const transactionForSign = await buildTxForApproveTradeWithRouter(swapParams.src);
console.log("Transaction for approve: ", transactionForSign);
```

Step 4: Confirm and Send the Transaction

```
const ok = await yesno({
  question: "Do you want to send a transaction to approve trade with 1inch router?"
});

if (!ok) {
  return false;
}

const approveTxHash = await signAndSendTransaction(transactionForSign);
console.log("Approve tx hash: ", approveTxHash);
```

After running this code, you'll receive a transaction hash. You can monitor its execution using the blockchain explorer.

Making the Swap

Before proceeding with the swap, please confirm that your approval transaction has a status of "Success."

Step 1: Set up your environment (same as before)

Step 2: Implement Helper Functions (same as before)

Step 3: Build the Body of the Transaction

```
async function buildTxForSwap(swapParams) {
  const url = apiRequestUrl("/swap", swapParams);

  // Fetch the swap transaction details from the API
  return fetch(url, headers)
    .then((res) => res.json())
    .then((res) => res.tx);
}

const swapTransaction = await buildTxForSwap(swapParams);
console.log("Transaction for swap: ", swapTransaction);
```

Step 4: Confirm and Send the Transaction

```
const ok = await yesno({
  question: "Do you want to send a transaction to exchange with 1inch router?"
});

if (!ok) {
  return false;
}

const swapTxHash = await signAndSendTransaction(swapTransaction);
console.log("Swap tx hash: ", swapTxHash);
```

After running this code, you'll receive a swap transaction hash. You can monitor its execution using the blockchain explorer.

Congratulations! You have successfully used the 1inch Swap API to perform a token swap at the best possible rate on the market. Always remember to double-check your transactions and use caution when dealing with your wallet's private key. Happy swapping!

How to Use the 1inch Swap API in Python

Step 1: Install the required packages

💡 If you have python ≥v3 installed you can use pip3 otherwise you can use pip install

```
pip3 install requests web3
```

Step 2: Set up your environment

Replace “[YOUR_API_KEY]” with the API key found here.

```
import requests
from web3 import Web3

chainId = 56   # Chain ID for Binance Smart Chain (BSC)
web3RpcUrl = "https://bsc-dataseed.binance.org"   # URL for BSC node
headers = { "Authorization": "Bearer [YOUR_API_KEY]", "accept": "application/json" }
walletAddress = "0x...xxx"   # Your wallet address
privateKey = "0x...xxx"   # Your wallet's private key. NEVER SHARE THIS WITH ANYONE!
```

Step 3: Define your swap parameters

```
swapParams = {
    "src": "0x111111111117dc0aa78b770fa6a738034120c302",   # Token address of 1INCH
    "dst": "0x1af3f329e8be154074d8769d1ffa4ee058b1dbc3",   # Token address of DAI
    "amount": "100000000000000000",   # Amount of 1INCH to swap (in wei)
    "from": walletAddress,
    "slippage": 1,   # Maximum acceptable slippage percentage for the swap (e.g., 1 for 1%)
    "disableEstimate": False,   # Set to True to disable estimation of swap details
    "allowPartialFill": False,   # Set to True to allow partial filling of the swap order
}
```

Step 4: Define API URLs and initialize Web3 libraries (Optional if not using Web3)

```
apiBaseUrl = f"https://api.1inch.dev/swap/v6.0/{chainId}"
web3 = Web3(web3RpcUrl);
```

Step 5: Define Helper Functions

```
# Construct full API request URL
def apiRequestUrl(methodName, queryParams):
    return f"{apiBaseUrl}{methodName}?{'&'.join([f'{key}={value}' for key, value in queryParams.items()])}"

# Function to check token allowance
def checkAllowance(tokenAddress, walletAddress):
    url = apiRequestUrl("/approve/allowance", {"tokenAddress": tokenAddress, "walletAddress": walletAddress})
    response = requests.get(url, headers=headers)
    data = response.json()
    return data.get("allowance")

# Sign and post a transaction, return its hash
async def signAndSendTransaction(transaction, private_key):
    signed_transaction = web3.eth.account.signTransaction(transaction, private_key)
    return await broadCastRawTransaction(signed_transaction.rawTransaction)
```

```
# Prepare approval transaction, considering gas limit
async def buildTxForApproveTradeWithRouter(token_address, amount=None):
    # Assuming you have defined apiRequestUrl() function to construct the URL
    url = apiRequestUrl("/approve/transaction", {"tokenAddress": token_address, "amount": amount} if amount el
se {"tokenAddress": token_address})
    response = requests.get(url, headers=headers)
    transaction = response.json()

    # Estimate gas limit
    wallet_address = "YOUR_WALLET_ADDRESS"
    gas_limit = web3.eth.estimateGas(transaction, from_address=wallet_address)

    return {**transaction, "gas": gas_limit}
```

Step 6: Check Token Allowance

```
allowance = checkAllowance(swapParams["src"], walletAddress)
print("Allowance: ", allowance)
```

> **NOTE**
>
> If the output shows `Allowance: 0`, it means that the 1inch router does not have access to swap this token within your wallet.

Creating the Token Allowance (Approval) Transaction

Before the 1inch aggregation protocol can access tokens in your wallet, you need to create an approval transaction. This transaction specifies that the 1inch router is allowed to swap a specific amount of the token you choose.

> **CAUTION**
>
> Please be cautious as approval transactions require payment of a blockchain gas fee, which is deducted in the form of native tokens from your wallet.

Step 1: Set up your environment (same as before)

Step 2: Implement Helper Functions (same as before)

Step 3: Build the Body of the Transaction

```
# Prepare approval transaction, considering gas limit

transactionForSign = await buildTxForApproveTradeWithRouter(swapParams.src)
```

Step 4: Confirm and Send the Transaction

```
ok = input("Do you want to send a transaction to approve trade with 1inch router? (y/n): ")
if ok.lower() == "y":
    # Sign and send the transaction for approval
    approveTxHash = signAndSendTransaction(transactionForSign)
    print("Approve tx hash: ", approveTxHash)
```

After running this code, you'll receive a transaction hash. You can monitor its execution using the blockchain explorer.

Making the Swap

> ⚠ CAUTION
>
> Before proceeding with the swap, please confirm that your approval transaction has a status of "Success."

Step 1: Set up your environment (same as before)

Step 2: Implement Helper Functions (same as before)

Step 3: Build the Body of the Transaction

```
def buildTxForSwap(swapParams):
    url = apiRequestUrl("/swap", swapParams)
    swapTransaction = requests.get(url,  headers={'Authorization':  f'Bearer YOUR_API_KEY'}).json()["tx"]
    return swapTransaction
```

Step 4: Confirm and Send the Transaction

```
ok = input("Do you want to send a transaction to exchange with 1inch router? (y/n): ")
if ok.lower() == "y":
    # You need to implement the signAndSendTransaction function to sign and send the transaction
    # Replace the following line with your actual implementation
    swapTxHash = signAndSendTransaction(swapTransaction)
    print("Swap tx hash: ", swapTxHash)
```

After running this code, you'll receive a swap transaction hash. You can monitor its execution using the blockchain explorer.

Congratulations! You have successfully used the 1inch Swap API to perform a token swap at the best possible rate on the market. Always remember to double-check your transactions and use caution when dealing with your wallet's private key. Happy swapping!

Previous
‹ Introduction

Next

Find the best quote to swap with 1inch Router

© 2025 1inch Limited