Quickstart guide

How to Use the 1inch Transaction Gateway API with Python and Request

In this tutorial, we will guide you through the process of using the 1inch Transaction Gateway API with Python and the `requests` library. The Transaction Gateway API allows you to broadcast public and private transactions on the Ethereum network using raw transaction data.

Before we begin, ensure you have Python installed on your system. If you don't have the `requests` library installed, you can install it using `pip` by running the following command:

```
pip install requests
```

Now, let's get started with step-by-step instructions!

Step 1: Import the `requests` Library

First, we need to import the `requests` library, which allows us to make HTTP requests to the 1inch Transaction Gateway API.

```
import requests
```

Step 2: Define the API Base URL

Next, we'll define the base URL for the 1inch Transaction Gateway API. We'll use this URL to construct our API requests.

```
base_url = "https://api.1inch.dev/tx-gateway/v1.1/1"
headers = { "Authorization": "Bearer [YOUR_API_KEY]", "accept": "application/json" }
```

Step 3: Broadcast Public Transaction

To broadcast a public transaction, we'll use the `/broadcast` endpoint. We'll provide the raw transaction data as the request body.

```
def broadcast_public_transaction(raw_transaction):
    endpoint = "/broadcast"
    data = {
        "rawTransaction": raw_transaction
    }
    response = requests.post(base_url + endpoint, headers=headers, json=data)
    if response.status_code == 200:
        return response.json()["transactionHash"]
    else:
        print(f"Failed to broadcast public transaction. Status code: {response.status_code}")
        return None
```

Step 4: Broadcast Private Transaction

To broadcast a private transaction, we'll use the `/flashbots` endpoint. Again, we'll provide the raw transaction data as the request body.

```
def broadcast_private_transaction(raw_transaction):
    endpoint = "/flashbots"
    data = {
        "rawTransaction": raw_transaction
    }
    response = requests.post(base_url + endpoint, headers=headers, json=data)
    if response.status_code == 200:
        return response.json()["transactionHash"]
    else:
        print(f"Failed to broadcast private transaction. Status code: {response.status_code}")
        return None
```

Step 5: Broadcast a Raw Transaction via RPC

If you prefer to broadcast a raw transaction via RPC, we'll use the `/rpc` endpoint. Similar to the previous steps, we'll provide the JSON-RPC request as the request body.

```
def broadcast_raw_transaction_via_rpc(json_rpc_request):
    endpoint = "/rpc"
    data = json_rpc_request
    response = requests.post(base_url + endpoint, headers=headers, json=data)
    if response.status_code == 200:
        return response.json()["result"]
    else:
        print(f"Failed to broadcast transaction via RPC. Status code: {response.status_code}")
        return None
```

Step 6: Putting It All Together

Now, let's put everything together and use the functions to broadcast transactions on the Ethereum network.

```
if __name__ == "__main__":
    # Replace with the raw transaction data you want to broadcast
    raw_transaction_data = "0xf86..."

    # Step 3: Broadcast public transaction
    transaction_hash_public = broadcast_public_transaction(raw_transaction_data)
    print("Transaction Hash (Public):", transaction_hash_public)

    # Step 4: Broadcast private transaction
    transaction_hash_private = broadcast_private_transaction(raw_transaction_data)
    print("Transaction Hash (Private):", transaction_hash_private)

    # Step 5: Broadcast transaction via RPC
    # Replace with the JSON-RPC request you want to use
    json_rpc_request_data = {
        "jsonrpc": "2.0",
        "id": "string",
        "method": "eth_sendRawTransaction",
        "params": [raw_transaction_data]
    }
    transaction_hash_rpc = broadcast_raw_transaction_via_rpc(json_rpc_request_data)
    print("Transaction Hash (RPC):", transaction_hash_rpc)
```

That's it! You have successfully used the 1inch Transaction Gateway API with Python and the `requests` library to broadcast public and private transactions, as well as transactions via RPC on the Ethereum network. Happy coding!

Full Script

Here you can find the full script with consideration of the default RPS limit

```
import requests
import time

base_url = "https://api.1inch.dev/tx-gateway/v1.1/1"
headers = { "Authorization": "Bearer [YOUR_API_KEY]", "accept": "application/json" }

def broadcast_public_transaction(raw_transaction):
    endpoint = "/broadcast"
    data = {
        "rawTransaction": raw_transaction
    }
    response = requests.post(base_url + endpoint, headers=headers, json=data)
    if response.status_code == 200:
        return response.json()["transactionHash"]
    else:
        print(f"Failed to broadcast public transaction. Status code: {response.status_code}")
        return None

def broadcast_private_transaction(raw_transaction):
    endpoint = "/flashbots"
    data = {
        "rawTransaction": raw_transaction
```

```
    }
    response = requests.post(base_url + endpoint, headers=headers, json=data)
    if response.status_code == 200:
        return response.json()["transactionHash"]
    else:
        print(f"Failed to broadcast private transaction. Status code: {response.status_code}")
        return None

def broadcast_raw_transaction_via_rpc(json_rpc_request):
    endpoint = "/rpc"
    data = json_rpc_request
    response = requests.post(base_url + endpoint, headers=headers, json=data)
    if response.status_code == 200:
        print(response.json())
        return response.json()["result"]
    else:
        print(f"Failed to broadcast transaction via RPC. Status code: {response.status_code}")
        return None

if __name__ == "__main__":
    # Replace with the raw transaction data you want to broadcast
    raw_transaction_data = "0xf86..."

    # Step 3: Broadcast public transaction
    transaction_hash_public = broadcast_public_transaction(raw_transaction_data)
    print("Transaction Hash (Public):", transaction_hash_public)
    # sleep one second because of RPS limit
    time.sleep(1)

    # Step 4: Broadcast private transaction
    transaction_hash_private = broadcast_private_transaction(raw_transaction_data)
    print("Transaction Hash (Private):", transaction_hash_private)
    # sleep one second because of RPS limit
    time.sleep(1)

    # Step 5: Broadcast transaction via RPC
    # Replace with the JSON-RPC request you want to use
    json_rpc_request_data = {
        "jsonrpc": "2.0",
        "id": "string",
        "method": "eth_sendRawTransaction",
        "params": [raw_transaction_data]
    }
    transaction_hash_rpc = broadcast_raw_transaction_via_rpc(json_rpc_request_data)
    print("Transaction Hash (RPC):", transaction_hash_rpc)
```

Previous
‹ Introduction

Next
Broadcast public transaction ›