Add Fees To Swap

INFO

As of January 2025, you no longer need to use the Referral Program to set up a `referralAccount` and `referralTokenAccount` to collect fees from the swaps you provide to the end users.

Simply, just pass in any valid token account as the `feeAccount` parameter in the Swap API.

NOTE

You can still find information about the Referral Program.

The Referral Program is an open source program by Jupiter to provide referral fees for integrators who are integrating Jupiter Swap and Jupiter Limit Order. You can check out the code here to gain a better understanding of how it works.

Use Case

By default, there are **zero** protocol fees on Jupiter Swap. Integrators have the option to introduce a platform fee denoted in basis points, e.g. **20 bps** for **0.2%** of the token input or output.

Important Notes

- **Input mint or the output mint** on the swap for ExactIn.
- **Input mint ONLY** on the swap for ExactOut.
- Example, if you swap JUP to USDC, you cannot take fees in SOL, it has to be part of the swap.
- It does not support Token2022 tokens.
- Referral Program is no longer required.

▸ **Via Referral Program (No longer required for Swap API)**

1. Set up

You will need to complete the prerequisites and understanding of Environment Setup and Get Quote and Swap guide as this is reliant on the Swap API.

2. Set your fee in Quote

Setting your fee is simple, just add `platformFeeBps` parameter to the `/quote` endpoint.

In this example, we set `platformFeeBps` to `20` which equates to 0.2%.

```
const quoteResponse = await (
    await fetch(
        'https://lite-api.jup.ag/swap/v1/quote?
inputMint=So11111111111111111111111111111111111111112&outputMint=EPjI
    )
  ).json();

console.log(JSON.stringify(quoteResponse, null, 2));
```

3. Set your feeAccount in Swap

In the `/swap` endpoint, you will need to pass in the `feeAccount` parameter. The `feeAccount` is any token account that will receive the fees from the swap. Do ensure that the token account is initialized and is the correct mint to receive the fees in.

```
const swapResponse = await (
    await fetch('https://api.jup.ag/swap/v1/swap', {
        method: 'POST',
        headers: {
        'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            quoteResponse,
            userPublicKey: wallet.publicKey.toBase58(), // Pass in
actual referred user in production
            feeAccount: feeAccount,
        })
    })
).json();
```

4. Sign and send transaction

Finally, the user can sign the transaction and it can be submitted to the network to be executed. You can refer to the Send Swap Transaction guide to complete this step.

Create Token Account

To create a token account, you can use the following code or refer to Solana Cookbook.

- The code creates the transaction to create the token account and handles the transaction siging and sending.
- If the token account already exists, it will not create and might throw an error such as `Provided owner is not allowed`.

```
import { createAssociatedTokenAccount } from "@solana/spl-token";

const mintPubkey = new PublicKey(
    "JUPyiwrYJFskUPiHa7hkeR8VUtAeFoSYbKedZNsDvCN",
);

let ata = await createAssociatedTokenAccount(
    connection, // connection
    wallet, // fee payer
    mintPubkey, // mint
    wallet.publicKey, // owner of the token account
    // confirmOptions, // if you need to skip simulation and send
the transaction immediately
    // programId, // if you need to use a different token program
id such as token-2022
    // associatedTokenProgramId,
    // allowOwnerOffCurve, // if you need to allow the owner to be
off curve
);
console.log(`ATA: ${ata.toBase58()}`);
```

Edit this page