Filling and canceling Solana Fusion order

Filling an order

The example below shows how to fill an existing Fusion order on Solana using 1inch Solana Fusion SDK.

```
import { Address, FusionSwapContract, Sdk } from '@1inch/solana-fusion-sdk'
import { AxiosHttpProvider } from '@1inch/solana-fusion-sdk/axios'
import { Connection, Keypair, PublicKey, Transaction } from '@solana/web3.js' // v1

const authKey = '...'
const secretKey = Buffer.from('secret', 'base64') // replace with real secret key
const rpc = 'https://api.mainnet-beta.solana.com'

const connection = new Connection(rpc)
const resolver = Keypair.fromSecretKey(secretKey)

const sdk = new Sdk(new AxiosHttpProvider(), { baseUrl: 'https://api.1inch.dev/fusion', authKey, version: 'v1.0'
})
const contract = FusionSwapContract.default()

const activeOrders = await sdk.getActiveOrders()
const orderToFill = activeOrders[0]

// fill order instruction
const ix = contract.fill(
  orderToFill.order,
  orderToFill.remainingMakerAmount, // fill for remaining amount
  accounts: {
      taker: Address.fromPublicKey(resolver.publicKey),
      maker: orderToFill.maker
      srcTokenProgram: Address.TOKEN_PROGRAM_ID,
      dstTokenProgram: Address.TOKEN_PROGRAM_ID,
  }
)

// generate tx
const fillTx = new Transaction().add({
  // at the moment of `ix` execution, resolver must have enough source tokens, so they can be transferred to user
  ...ix,
  programId: new PublicKey(ix.programId.toBuffer()),
  keys: ix.accounts.map((a) => ({
    ...a,
    pubkey: new PublicKey(a.pubkey.toBuffer())
  }))
})

fillTx.recentBlockhash = (await connection.getRecentBlockhash()).blockhash
fillTx.sign(resolver)

// broadcast tx
await connection.sendRawTransaction(createOrderTx.serialize())
```

Canceling an order

The example below shows how to cancel a Fusion order on Solana using the 1inch Fusion SDK. For canceling expired orders, a resolver will be paid `order.resolverCancellationConfig.maxCancellationPremium` lamports.

```
import { Address, FusionSwapContract, Sdk } from "@1inch/solana-fusion-sdk";
import { AxiosHttpProvider } from "@1inch/solana-fusion-sdk/axios";
import { Connection, Keypair, PublicKey, Transaction } from "@solana/web3.js"; // v1

const authKey = "...";
const secretKey = Buffer.from("secret", "base64"); // replace with real secret key
const rpc = "https://api.mainnet-beta.solana.com";

const connection = new Connection(rpc);
const maker = Keypair.fromSecretKey(secretKey);

const sdk = new Sdk(new AxiosHttpProvider(), {
  baseUrl: "https://api.1inch.dev/fusion",
  authKey,
  version: "v1.0",
});

const cancellableOrders = await sdk.getOrdersCancellableByResolver();
const order = cancellableOrders[0];
const contract = FusionSwapContract.default();

// close escrow instruction
const ix = contract.cancelOrderByResolver(order.order, {
  maker: order.maker,
  resolver: Address.fromPublicKey(resolver.publicKey),
  srcTokenProgram: Address.TOKEN_PROGRAM_ID,
});

// generate tx
const cancelOrderTx = new Transaction().add({
  ...ix,
  programId: new PublicKey(ix.programId.toBuffer()),
  keys: ix.accounts.map((a) => ({
    ...a,
    pubkey: new PublicKey(a.pubkey.toBuffer()),
  })),
});
```

```
cancelOrderTx.recentBlockhash = (
  await connection.getRecentBlockhash()
).blockhash;
cancelOrderTx.sign(resolver);

// broadcast tx
await connection.sendRawTransaction(createOrderTx.serialize());
```