Introduction

The limit-order-sdk is a TypeScript library developed by 1inch that enables developers to build, sign, and manage off-chain limit orders, fully compatible with the 1inch Limit Order Protocol.

This SDK handles the technical details of building EIP-712 compliant orders and preparing them for on-chain execution. You can use it to easily create orders off-chain and send them to the 1inch Orderbook API or fill them directly on-chain.

With Limit Order SDK, you can:

- Build limit orders
- Sign orders using EIP-712 format
- Use permits for gasless approvals
- Support trades with ERC-20, ERC-721, and ERC-1155 tokens

How this SDK fits into 1inch ecosystem

Component	What it does
Limit Order Protocol	Smart contracts that check and fill orders on-chain
Orderbook API	API for storing and sharing signed orders
limit-order-sdk	Tool to build and sign valid orders before sending them

Getting started

Install the SDK

Install the SDK using npm:

```
npm install @1inch/limit-order-sdk
```

Requirements

Before using the SDK, make sure your project includes:

- Node.js (v14 or newer)
- A supported Web3 provider (e.g. ethers.js or web3.js)
- TypeScript support (recommended)
- A wallet or private key to sign orders

Web3 provider

The SDK requires a Web3 provider—a connection to the blockchain. You can use one from `ethers`, `web3`, or your app's environment. Example using `ethers`:

```
import { providers } from "ethers";

const provider = new providers.JsonRpcProvider(
  "https://mainnet.infura.io/v3/YOUR_INFURA_KEY",
);
```

Example: Build and sign a limit order

```
import {
  LimitOrder,
  MakerTraits,
  Address,
  Sdk,
  randBigInt,
  FetchProviderConnector,
} from "@1inch/limit-order-sdk";
import { Wallet } from "ethers";

// it is a well-known test private key, do not use it in production
const privKey =
  "0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80";
const authKey = "...";
const maker = new Wallet(privKey);
const expiresIn = 120n; // 2m
const expiration = BigInt(Math.floor(Date.now() / 1000)) + expiresIn;

const UINT_40_MAX = (1n << 48n) - 1n;

// see MakerTraits.ts
const makerTraits = MakerTraits.default()
  .withExpiration(expiration)
  .withNonce(randBigInt(UINT_40_MAX));

const sdk = new Sdk({
  authKey,
  networkId: 1,
  httpConnector: new FetchProviderConnector(),
});
```

```
const order = await sdk.createOrder(
  {
    makerAsset: new Address("0xdac17f958d2ee523a2206206994597c13d831ec7"),
    takerAsset: new Address("0x111111111117dc0aa78b770fa6a738034120c302"),
    makingAmount: 100_000000n, // 100 USDT
    takingAmount: 10_000000000000000000n, // 10 1INCH
    maker: new Address(maker.address),
    // salt? : bigint
    // receiver? : Address
  },
  makerTraits,
);

const typedData = order.getTypedData();
const signature = await maker.signTypedData(
  typedData.domain,
  { Order: typedData.types.Order },
  typedData.message,
);

await sdk.submitOrder(order, signature);
```

Advanced usage

In addition to publishing limit orders to the 1inch Orderbook, the Limit Order SDK can also be used for more advanced and customized workflows that operate independently of the Orderbook API. To learn more about this use case, see Advanced usage for custom protocol integration.

Previous
‹ Get calculated making amount on trading pair by provided amount

Next
Limit orders ›