Limit orders

Limit orders let a user define a fixed price and amount for a token trade. These orders are:

- Created and signed off-chain
- Shared through APIs (like the 1inch Orderbook API)
- Filled on-chain through the Limit Order Protocol

Creating and signing a limit order

Example

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

import { Wallet } from "ethers";

// Example private key (test only — do not use in production!)
const privKey =
  "0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80";
const maker = new Wallet(privKey);

const sdk = new Sdk({
  authKey: "YOUR_AUTH_KEY", // Get this from 1inch Developer Portal
  networkId: 1,
  httpConnector: new FetchProviderConnector(),
});

// Optional expiration setup
const expiresIn = 120n; // 2 minutes
const expiration = BigInt(Math.floor(Date.now() / 1000)) + expiresIn;
const UINT_40_MAX = (1n << 48n) - 1n;

const makerTraits = MakerTraits.default()
  .withExpiration(expiration)
  .withNonce(randBigInt(UINT_40_MAX));
```

```
const order = await sdk.createOrder(
  {
    makerAsset: new Address("0xdAC17F958D2ee523a2206206994597C13D831ec7"), // USDT
    takerAsset: new Address("0x111111111117dc0aa78b770fa6a738034120c302"), // 1INCH
    makingAmount: 100_000000n, // 100 USDT (6 decimals)
    takingAmount: 10_00000000000000000n, // 10 1INCH (18 decimals)
    maker: new Address(maker.address),
  },
  makerTraits,
);

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

// (Optional) Submit to 1inch Orderbook API
await sdk.submitOrder(order, signature);
```

Signing

The order is signed using EIP-712 structured data format. This guarantees that it can only be filled exactly as requested by the maker.

You don't need to build typed data manually; `order.getTypedData()` handles that for you.

`MakerTraits` and expiration

`MakerTraits` is a utility that lets you define how the order behaves. Use it to configure:

- Whether the order can be partially filled
- Whether it can be filled multiple times
- A custom expiration timestamp
- A unique nonce

Usage example:

```
const traits = new MakerTraits()
  .withPartialFill()
  .withExpiration(expiration)
  .withNonce(randBigInt(UINT_40_MAX));
```

Order fields

Field	Type	Description
`makerAsset`	`string`	Token the maker is selling

Field	Type	Description
`takerAsset`	`string`	Token the taker is expected to pay
`makingAmount`	`bigint`	Amount the maker is offering
`takingAmount`	`bigint`	Amount the maker wants in return
`maker`	`string`	Address that signs the order (usually a wallet)

Submitting to the Orderbook (optional)

If you want your order to be visible and fillable by professional resolvers, send it to the Orderbook API. This will require using an authentication key. You can get the key from the 1inch Developer Portal.

Previous
‹ Introduction

Next
Advanced usage for custom protocol integration ›