Introduction

Limit Order Protocol

The key features of the protocol are extreme flexibility and high gas efficiency, which are achieved with the following features

Basic features

- Select an asset receiver for an order.
- Choose whether to allow or disallow partial and multiple fills.
- Define conditions that must be met before execution can proceed (e.g. stop-loss, take-profit orders).
- Specify interactions (arbitrary maker's code) to execute before and after order filling.
- Choose an approval scheme for token spend (approve, permit, permit2).
- Request that WETH be unwrapped to ETH either before (to sell ETH) or after the swap (to receive ETH).
- Make an order private by specifying the only allowed taker's address.
- Set the order's expiration date.
- Assign a nonce or epoch to the order for easy cancellation later.

Advanced features

- Define a proxy to handle transfers of assets that are not compliant with `IERC20`, allowing the swapping of non-ERC20 tokens, such as ERC721 or ERC1155.
- Define functions to calculate, on-chain, the exchange rate for maker and taker assets. These functions can be used to implement dutch auctions (where the rate decreases over time) or range orders (where the rate depends on the volume already filled), among others.

RFQ orders

Separate RFQ order are deprecated in v4. To create the most gas efficient order use a basic order without extensions.

Supported tokens

- ERC 20
- ERC 721
- ERC 1155
- Other token standards could be supported via external extension

Functions

constructor

```
constructor(contract IWETH _weth) public
```

DOMAIN_SEPARATOR

```
function DOMAIN_SEPARATOR() external view returns (bytes32)
```

Returns the domain separator for the current chain (EIP-712)

pause

```
function pause() external
```

Pauses all the trading functionality in the contract.

unpause

```
function unpause() external
```

Unpauses all the trading functionality in the contract.

Previous
‹ IUniswapV3SwapCallback

Next
OrderLib ›