EasySwap 合约方案设计

需求

基于智能合约实现基于**订单簿模型**的NFT交易系统, 即能够支持以下写入操作和查询操作;

写入操作:

链上订单簿(OrderBook DEX)支持 create limit sell/buy, market sell/buy order, edit(cancel&create)/cancel order 功能;

1. **limit sell order:** 即list, 挂限价卖单; 可以针对单个collection的**指定item**;
    a. 资产转移: 将指定的item转移至dex合约中;
2. **limit buy order:** 即bid(make offer), 挂限价买单; 可以针对单个collection的**指定item**, 也可以针对**整一个collection**的所有item;
    a. 资产转移:
3. **market sell order:** 即接bid, 限价买单的对手单, 针对单个collection的单个item;
    a. 情况1: 用户利用自己拥有的nft, 接受指定的买单(bid);
    b. 情况2: 用户利用自己某个**有效卖单(list)或者“过期但没被fill”**中的nft, 接受指定的买单(bid);
4. **market buy order:** 即接list, 限价卖单的对手单, 针对单个collection的**单个item**;
5. **edit/cancel order:** 编辑订单信息: 价格和有效期; 取消订单, 即使订单失效;

查询操作:

1. 支持从链上查询订单（包括已经过期订单）;

合约构成及组件

1. OrderBook: 实现完整的订单簿交易逻辑
    a. OrderStorage: 用于**存储订单信息**的模块
    b. OrderValidator: 用于处理**订单逻辑验证**的模块
    c. ProtocolManager: 用于**管理协议费**的模块
    d. 版税管理。。。
    e. 。。。

2. OrderVault: 独立存储订单相关资产的模块;

主流程函数

```
interface IEasySwapOrderBook {
    /**
     * @notice Create multiple orders and transfer related assets.
     * @dev If Side=List, you need to authorize the EasySwapOrderBook contract first (creating a List order will transfer the NFT to the order pool).
     * @dev If Side=Bid, you need to pass {value}: the price of the bid (similarly, creating a Bid order will transfer ETH to the order pool).
     * @dev order.maker needs to be msg.sender.
     * @dev order.price cannot be 0.
     * @dev order.expiry needs to be greater than block.timestamp, or 0.
     * @dev order.salt cannot be 0.
     * @param newOrders Multiple order structure data.
     * @return newOrderKeys The unique id of the order is returned in order, if the id is empty, the corresponding order was not created correctly.
     */
    function makeOrders(
        LibOrder.Order[] calldata newOrders
    ) external payable returns (OrderKey[] memory newOrderKeys);

    /**
     * @notice Cancels multiple orders by their order keys.
     * @param orderKeys The array of order keys to cancel.
     * @return successes Array of boolean values indicating the success of each cancellation.
     */
    function cancelOrders(
        OrderKey[] calldata orderKeys
    ) external returns (bool[] memory successes);

    /**
     * @notice Cancels multiple orders by their order keys.
     * @dev newOrder's saleKind, side, maker, and nft must match the corresponding order of oldOrderKey, otherwise it will be skipped; only the price can be modified.
     * @dev newOrder's expiry and salt can be regenerated.
     * @param editDetails The edit details of oldOrderKey and new order info
     * @return newOrderKeys The unique id of the order is returned in order, if the id is empty, the corresponding order was not edit correctly.
     */
```

```
33      function editOrders(
34          LibOrder.EditDetail[] calldata editDetails
35      ) external payable returns (OrderKey[] memory newOrderKeys);
36
37      function matchOrder(
38          LibOrder.Order calldata sellOrder,
39          LibOrder.Order calldata buyOrder
40      ) external payable;
41
42      /**
43       * @dev Matches multiple orders atomically.
44       * @dev If buying NFT, use the "valid sellOrder order" and construct a
    matching buyOrder order for order matching:
45       * @dev    buyOrder.side = Bid, buyOrder.saleKind = FixedPriceForItem,
    buyOrder.maker = msg.sender,
46       * @dev    nft and price values are the same as sellOrder,
    buyOrder.expiry > block.timestamp, buyOrder.salt != 0;
47       * @dev If selling NFT, use the "valid buyOrder order" and construct a
    matching sellOrder order for order matching:
48       * @dev    sellOrder.side = List, sellOrder.saleKind = FixedPriceForItem,
    sellOrder.maker = msg.sender,
49       * @dev    nft and price values are the same as buyOrder,
    sellOrder.expiry > block.timestamp, sellOrder.salt != 0;
50       * @param matchDetails Array of `MatchDetail` structs containing the
    details of sell and buy order to be matched.
51       * @return successes Array of boolean values indicating the success of
    each match.
52       */
53      function matchOrders(
54          LibOrder.MatchDetail[] calldata matchDetails
55      ) external payable returns (bool[] memory successes);
56  }
```

事件

事件声明

```
1   event LogMake( // topic0:
    0x4bf66e804e33459ec573a8989a7aad0bcc356347bec7ad581924e6399a7a827e
2       bytes32 orderKey,
3       uint8 indexed side,
4       uint8 indexed saleKind,
5       address indexed maker,
```

```
    LibOrder.Asset nft,
    Price price,
    uint64 expiry,
    uint64 salt
);

event LogCancel(bytes32 indexed orderKey, address indexed maker);

event LogMatch(
                OrderKey indexed makeOrderKey,
    OrderKey indexed takeOrderKey,
                LibOrder.Order makeOrder,
    LibOrder.Order takeOrder,
    Price priceFilled
);

struct Asset {
    address collection;
    uint256 tokenId;
}
type Price is uint128;

struct Order {
    Side side;
    SaleKind saleKind;
    address maker;
    Asset nft;
    Price price;
    uint64 expiry;
    uint64 salt;
}
```

事件样例参考

```
// LogMake样例
topics: [
  '0x4bf66e804e33459ec573a8989a7aad0bcc356347bec7ad581924e6399a7a827e',
  '0x0000000000000000000000000000000000000000000000000000000000000000',
  '0x0000000000000000000000000000000000000000000000000000000000000001',
  '0x000000000000000000000000f39fd6e51aad88f6f4ce6ab8827279cfffb92266'
],
data:
'0x01d919ebb6c257a9bbe0f96a4edd9ec1677f1ec1ce51b8a28105b09d1e6a9b8000000000000
```

```
000000000000059b670e9fa9d0a427751af201d676719a970857b0000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
00000000002386f26fc10000000000000000000000000000000000000000000000000000000000
0006544e728000000000000000000000000000000000000000000000000000000000000000001',
```

```
//LogCancel
topics: [
  '0x0ac8bb53fac566d7afc05d8b4df11d7690a7b27bdc40b54e4060f9b21fb849bd',
  '0x7ccab99702f5755f281db6846523e1bf738e279bab3489408001022f5e85d2aa',
  '0x000000000000000000000000f39fd6e51aad88f6f4ce6ab8827279cfffb92266'
],
```

```
//LogMatch
topics: [
  '0xc8009471c7c8fc4ad318bf0b9d4091059119ef000147915e5500ae341c5baaf5',
  '0xa57a418e023548993d23700718ff9f48b967c8c3d5c90896276b3939f5255a73',
  '0x0db709770420d71816fc48963b7d5ea275e47b2cfbaa267406ba07031aacd794'
],
data:
'0x0000000000000000000000000000000000000000000000000000000000000100000000000
0000000000000000000000000000000000000000000000000000000000000100000000000000000000000070
0997970c51812dc3a010c7d01b50e0d17dc79c8000000000000000000000000922d6956c99e12d
feb3224dea977d0939758a1fe0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000470de4df820000000
0000000000000000000000000000000000000000000000006544f5820000000000000000000
0000000000000000000000000000000000000000000000020000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000010000000000000000000000f39fd6e51aad88f6f4ce6ab8827279cfffb
92266000000000000000000000000922d6956c99e12dfeb3224dea977d0939758a1fe000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000002386f26fc10000000000000000000000000000000000000000000000000000
0000000000000000006544f5820000000000000000000000000000000000000000000000000000000000000000
0000000000010000000000000000000000000000000000000000000002386f26fc10000'
,
```