CoW AMM Liquidity for Solvers

I'm a solver. How do I use CoW AMM liquidity?

CoW AMM orders already appear in the CoW Protocol orderbook, so you're already using its liquidity. However, CoW AMMs allow solvers to specify custom buy and sell amounts, as long as the order preserves or increase the constant product invariant of the token reserves.

CoW AMMs can be treated as a liquidity source akin to Uniswap or Balancer weighted pools with uniform weights. Each CoW AMM is a pair that trades two tokens.

Importantly, surplus for a CoW AMM order is measured differently when computing the solver reward payout.

Indexing Balancer CoW AMMs

CoW AMM pairs can be detected by listening to theCOWAMMPoolCreated events emitted by theBCowFactory instance for the current chain (see theofficial contract repository for the current deployment addresses). All addresses of Balancer CoW AMMs are sent as part of the auction instance assurplusCapturingJitOrderOwners .

The AMM reserves are the balance of the two tokens traded by the AMM.

Settling a custom order

You need to choose a valid CoW Swap order with the following restrictions:

- sellToken
- : any token in the pair.
- buyToken
- : the other token in the pair.
- receiver
- : must beRECEIVER_SAME_AS_OWNER
- (zero address).
- sellAmount
- : any value.
- buyAmount
- : any value such that, after trading these exact amounts, the product of the token reserves is no smaller than before trading.
- validTo
- : at most 5 minutes after the block timestamp at execution time.
- appData
- : must be the value specified instaticInput
- .
- feeAmount
- : must be zero.
- kind
- : any value.
- partiallyFillable
- : any value.
- sellTokenBalance
- : must beBALANCE_ERC20
- .
- buyTokenBalance
- : must beBALANCE_ERC20
- .

note The sell and buy amount specified are the limit amounts of the order. The actual executed amounts are determined using prices from the auction. In that regard, CoW AMM orders behave exactly like normal trader orders. You also need to compute:

- the order hashhash
- as defined in the libraryGPv2Order
- , and
- the order signature

Example code that generates an order can be found intheorder function of the BCoWHelper contract.

This order can be included in a batch as any other CoW Protocol orders with two extra conditions:

- One of the pre-interactions must set the commitment by callingBCoWPool.commit(hash)
- .
- Must contain at most one order from the AMM in the same batch.

Solvers using the non co-located driver should add aJitTrade and apreInteraction to their solution.

::

Surplus

The surplus for a CoW AMM order is measured the same way as other orders, by comparing the traded amounts to limit amounts.

When creating a CoW AMM order it is therefore encouraged to use the smallest possible limit amount which does not violate the invariant of the pool.

If we call$X$ (resp.Y) the reserves of sell (resp. buy) token, and$x$ (resp.y) the executed sell (resp. buy) amount, then the minimal buy amount is$x Y / (X - x)$ and the order surplus with that choice for the limit amount is:

$$\text{surplus} = y - \frac{x Y}{X - x} .$$

Maximizing this quantity will lead to the largest score in the solver competition. Edit this page
Previous Deploying a CoW AMM Using BalancerNext Governance