Solver competition accounting process

There is an accounting process in place for the solver competition. It is performed separately on each chain. Currently, the execution of the accounting process takes place once a week for all chains the protocol operates on, and each accounting week starts on Tuesday at midnight UTC and concludes a week later at midnight UTC. This accounting process consists of the following:

- calculation of total rewards/penalties, as specified by themechanism
- ;
- calculation of protocol and partner fees;
- settlement contract buffers accounting, which consists of (1) calculation of protocol and partner fees that solvers might deposit in the settlement contract, (2) calculation of network fees (meant to cover gas) that solvers might deposit in the settlement contract, and (3) all other imbalances appearing in the settlement contract after a transaction gets executed, for solvers that choose the settlement contract as their execution layer.

A summary of the above, that is used to verify the accounting process each week, can be found in this Dune dashboard:https://dune.com/cowprotocol/cow-solver-rewards .

Auction rewards and penalties

Each auction with a winner has a reward/penalty associated with it, that is computed after the auction deadline passes. In the case of a successful submission on-chain within the auction deadline, a reward is computed, originally in the native token of the chain, as determined by themechanism . On the other hand, in the case of an unsuccessful execution (e.g., revert or delayed execution), a penalty is computed, originally expressed in the native token of the chain, that again is determined by the mechanism.

In each accounting week, we first identify all auctions that took place within that week. Specifically:

- we identify all blocks with a timestamp that is at least as large as the starting Tuesday, 00:00 UTC timestamp and strictly smaller than the ending Teusday, 00:00 UTC timestamp (this Dune query
- does exactly this computation). Let [X,Y] denote this interval;
- we then look at all auctions whose block deadline (i.e., the latest block for which the on-chain submission is considered valid/on-time) is larger or equal than X and smaller or equal than Y, and these are the auctions for which we compute rewards for that particular accounting week.

At the end of each accounting week, for each solver, the rewards and penalties are aggregated, and we have a performance reward per solver; note this can be negative in the case where penalties exceed rewards. The reward, naturally expressed in the native token, for each auction can be found in this Dune table:https://dune.com/queries/4351957 (seecapped_payment column).

Moreover, for each order executed on-chain, the solver that provided the quote that led to the order creation is rewarded, as determined by the mechanism of the price estimation competition. The solver that provided the winning quote for each order can be found in this Dune table:https://dune.com/queries/4364122 (seequote_solver column). We clarify here that for quote rewards, we consider all orders that got executed in a block that is larger or equal to X and smaller or equal to Y, where X and Y are defined as above. Note that this is slightly different compared to the auctions considered in the same time interval, as we use block deadlines for auctions while execution blocks for orders.

We stress that performance rewards and quote rewards are kept separate in the accounting.

We also highlight that performance and quote rewards are both paid in COW. To convert rewards from the native currency to COW, we use the average COW price, in USD, as provided in theprices.usd Dune table, where the blockchain is set to Ethereum, so as to ensure that this price is always available. The average is taken over the last 24h of the accounting period, in order to reduce the possibility of manipulating that price (seehere for the Dune sql code that does this specific calculation). The native token's average price in USD is also computed similarly (seehere). Once these two prices are available, the conversion from native token to COW and vice versa is straightforward.

Protocol and partner fees

Certain orders come with a list of so-called fee policies; these policies provide conditions about when an order is supposed to pay a protocol/partner fee, and if an order should pay a fee, then they also determine the fee amount as well. We stress that fees that a trade should pay can be fully determined by knowing the fee policies of the order and by observing the on-chain execution. For every CoW Protocol trade, one can check what protocol and partner fees were charged to the order (if any), by searching via the api for the corresponding order (see here:https://api.cow.fi/docs/#/default/get_api_v1_trades). We note that from a solver point of view, protocol and partner fees are treated uniformly, and the only thing that differentiates them is the recipient of the fee; in the first case it is the CoW DAO, while in the second case it is the corresponding partner.

Protocol and partner fees are naturally denominated in the surplus token of an order, and the accounting process uses the

native prices provided in each auction to convert these fee amounts to the native token of the chain; this implies that the exchange rate of these fees with respect to the native token of the chain is determined at auction creation time.

For simplicity, the core team also maintains the following Dune table (https://dune.com/queries/4364122), that among other things, reveals the amounts charged as protocol and partner fees on a per trade basis. The relevant columns areprotocol_fee andpartner_fee . Note that theprotocol_fee entry is the total protocol and partner fee charged, so in case there is a non-zero partner fee, in order to determine what amount is meant to be sent to the CoW DAO, one needs to subtract thepartner_fee entry from theprotocol_fee entry. The columnprotocol_fee_native_price can then be used to determine how these fee amounts are converted to the native token of the chain on a per trade basis (note that in order to get the final amount in the native token, one needs to multiply with theprotocol_fee_native_price and then divide by 10^18).

Buffer accounting

As mentioned above, buffer accounting consists of:

- protocol and partner fees that solvers might deposit to the settlement contract,
- network fees that solvers might deposit to the settlement contract, and
- all other imbalances (after protocol/partner/network fees have been deducted) appearing in the settlement contract after a transaction gets executed, for solvers that choose the settlement contract as their execution layer; we call these imbalances slippage.

Protocol and partner fees in the settlement contract

Solvers can decide to collect protocol and partner fees in the settlement contract, in order to then redirect these funds to the DAO or the relevant partner as protocol or partner fees, respectively. These amounts are converted to the native token of the chain by using the prices provided in the auction instance where each order got executed. If a solver decides to collect these fees in the settlement contract, then during the accounting, these fee amounts are converted to the native token, as explained above, and are sent by the main accounting transaction to the appropriate receiver.

As noted, by knowing the fee policies associated with an order and observing the on-chain execution of an order, one can fully determine the protocol fees associated with the execution. As a reminder, the following Dune table (https://dune.com/queries/4364122), among other things, reveals the amounts charged as protocol and partner fees on a per trade basis, and it is one of the main tables that are used in the buffer accounting process on Dune.

Network fees in the settlement contract

Similar to protocol fees, solvers can decide to collect part of the sell amount of an order in the settlement contract, as a network fee that would cover the gas cost associated with the execution of an order. These fees, specified in the sell token, are converted to the native token of the chain, using the prices provided in the auction instance where each order got executed, and are sent to solvers at the end of each accounting week. There is a dedicated table maintained by the core team that exposes the network fees (as well as the protocol fees) charged to each trade, and can be foundhere .

Implementation details

In this section, we share a few details about how protocol/partner/network fees are computed by observing an on-chain execution. When it comes to protocol and partner fees, one needs to be aware of the fee policies associated with an order. These fee policies are shared with solvers as part of the auction instance sent to them, and are also revealed for each trade in the api (see here:https://api.cow.fi/docs/#/default/get_api_v1_trades). This allows us to fully determine the protocol and partner fees associated with the order.

To use an example, suppose that we have a sell order that sells 1 WETH and the user receives 3000 USDC. When checking the api, we might see the following:

"executedProtocolFees": [{ "policy": { "surplus": { "factor": 0.5, "maxVolumeFactor": 0.01 } }, "amount": "5000000", "token": "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48" }] The above suggests that the protocol fee associated with the order was 5 USDC.

Once we know how much is the protocol fee, we can add it to the amount received by the user, in case of a sell order, or subtract it from the sell amount of the trade, in case of a buy order, in order to compute what we call the "raw buy amount" (or "raw sell amount" respectively). Using this amount and the uniform clearing price vector the solver proposed in its execution, we can now identify the network fee the solver charged the user.

To elaborate a bit on the above, we clarify that for each transaction, a solver is required to report a uniform clearing price vector as part of the calldata, meaning a vector that has one price per token traded, as well as custom prices on a per trade basis. These custom prices allow a solver to capture fees, whenever needed. The uniform clearing price vector is meant to describe the exchange rates in the absence of fees. So once we have the raw sell/buy amount, we can apply the uniform clearing price vector and see how much the user "would" have sold if there was no network fee charged. The difference between what the user actually sold and what they would have sold in case of zero network fees exactly identifies how much was the network fee charged to the trade.

Going back to our example, suppose the solver charged 0.001 WETH for the above trade. This means that the solver received 1 WETH, kept 0.001 WETH aside, and then swapped 0.999 WETH for 3005 USDC. The solver then realized that the protocol fee is 5 USDC, and thus kept 5 USDC aside, while returning 3000 USDC to the user. The uniform clearing prices, since they capture the exchange rate in the absence of fees, would then be set as follows: ucp(WETH) = 3005 and ucp(USDC) = 0.999. While the custom trade prices would be set to p(WETH) = 3000 and p(USDC) = 1.

From the actual on-chain execution, as we mentioned, one can reverse-engineer the solver's execution and determine using the fee policies that the protocol fee is 5 USDC. Which means that the raw buy amount is equal to 3005 USDC. By observing the calldata, we recover the UCP vector, and apply the exchange rate implied by that vector, which gives that the user "should" have sold 3005 * 0.999 / 3005 = 0.999 WETH. Since the user actually sold 1 WETH, we then conclude that the difference, i.e., 1 - 0.999 = 0.001 WETH, was the network fee the solver charged for the trade.

Slippage

Slippage accounting, that is taking place in all chains the protocol operates on, is performed on a per settlement/transaction basis, and the main query that executes it can be found here:https://dune.com/queries/4070065 . Roughly speaking, for each transaction executed on-chain, we compute the raw token imbalances of the settlement contract (i.e., we look at the state of the contract before and after execution, and the difference is what we call token imbalances), we account for protocol, partner and network fees (as described in the previous sections) in the cases where solvers deposit them in the settlement contract, and after we subtract those, the remaining imbalances (if any) are converted to the native token of the chain by using some price feed.

The price feed used for the accounting is constructed in this queryhttps://dune.com/queries/4064601 . In a few words, we use theprices.usd table of Dune and for a given imbalance on token X that a certain transaction caused, we take the average price of the token in a 1h-interval around the time of the trade, and use this price to evaluate the imbalance. In case the token is missing from the Dune table, we then resort to computing an average price of the token around the time of the trade by using other CoW Protocol trades that sell or buy this specific token, where the other side of the trade is a token with a price in theprices.usd table. If none of these succeed, then there is no price associated with the token in the price feed.

Slippage accounting details

We now discuss some more details. To properly do slippage accounting, we start by computing raw imbalances caused by a transaction. The raw imbalance vector of a transaction is defined as follows. Letbuffers(0) denote the vector that describes all balances the settlement contract holds just before the execution of a transaction. Letbuffers(1) denote the vector of all balances the settlement contract holds just after the execution of a transaction. The difference of the two vectors, i.e.,buffers(1) - buffers(0) , is what we call raw imbalances. This is computed in this query on a per transaction basis:https://dune.com/queries/4021644 .

Once we get the raw imbalances, the "fee corrections" are added for each transactionhttps://dune.com/queries/4059683 . Specifically, if we expect the solvers to deposit network, protocol and partner fees in the contract, and since these use the price feed provided in the corresponding auction instance and not the Dune price feed, what we do is that we assume that these fees were deposited fully in the settlement contract, and thus they need to be subtracted from the raw imbalances in order to evaluate how much "left-over" imbalance is there. This "left-over" imbalance is what we call slippage and this is what is evaluated using the price feed constructed inhttps://dune.com/queries/4064601 .

We note that currently all solvers use the settlement contract to deposit protocol/partner/network fees, and these fees are delivered to the appropriate receiver each Tuesday via the main accounting script the core team maintains. Edit this page
Previous Solver rewards Next Schemas