Solver rewards

The protocol is currently subsidizing the solver competition on Ethereum Mainnet (but not on Gnosis Chain), by rewarding solvers on a weekly basis (currently, every Tuesday) with rewards paid in COW. Solvers are rewarded based on their performance as solvers (i.e., when participating in the standard solver competition) as specified byCIP-20 andCIP-36 , and by participating in the price estimation competition and providing quotes that are needed for the gas estimates and limit price computations of market orders, as specified byCIP-27 andCIP-36 .

The annual rewards budget as of now is 16M COW, and the intended split between solver competition and price estimation rewards is currently at 80% for the solver competition (12.8M COW) and 20% for the price estimation competition (3.2M COW).

note For the interested reader, the main source of truth for the weekly payments to solvers is thisDune dashboard . The dashboard is populated with data aggregated by scripts within thesolver-rewards anddune-sync repositories.

Solver competition rewards (CIP-20 & CIP-36)

As specified inCIP-20 , solver competition rewards are split into two components: per-auction rewards and consistency rewards.

The auction's goal is to reward solvers based on how much value they provide to the users. In each auction cycle, solvers compete by proposing a solution to settle the batch, and the winning solver receives a reward that is part inETH and part inCOW (see below).

note The reward in ETH is not part of the rewards budget, and is meant to cover the costs associated with the gas spent by each solver; importantly, this ETH is obtained by converting the gas fees user orders pay and so have nothing to do with the COW rewards budget. As already mentioned, CoW Protocol has committed to spending a specified quantity ofCOW tokens to reward solvers. An estimated 12.8M COW annual budget is currently allocated for the solver competition rewards, which corresponds to an average weekly budget of ~250k COW. The part of the rewards budget inCOW that is not spent on per-auction rewards is used for consistency rewards. Consistency rewards are distributed weekly to each solver in proportion to the number of valid solutions submitted during the previous week.

Specification of per-auction rewards

The per-auction rewards are computed using a mechanism akin to a second-price auction. First, each solver commits to a numerical score and a solution (which includes a price vector and a list of trades to execute). The solver with the highest score wins the right to settle their submitted solution on chain.

note From the protocol's perspective, the solution executed on chain must equal the solver's initial commitment. The payment to the winning solver is

payment

cap (observedQuality − referenceScore) . \textrm{payment} = \textrm{cap}(\textrm{observedQuality} - \textrm{referenceScore}). payment

= cap (observedQuality

− referenceScore) . Here,referenceScore \textrm{referenceScore} referenceScore refers to the second-highest submittedscore \textrm{score} score , andobservedQuality \textrm{observedQuality} observedQuality denotes the settlement's quality as observed on chain. More precisely, in case of a successful settlement, theobservedQuality \textrm{observedQuality} observedQuality is equal to the sum of the surplus generated for users and fees paid to the protocol, while in the case of a failed settlement (e.g., one that reverted), theobservedQuality \textrm{observedQuality} observedQuality is zero.

note The payment calculation can result in a negative figure, in which case the solver is required to pay the amount to the protocol. The payment is capped from above and below using the functioncap (x) = max (− c l , min (c u + observedCost , x)) \textrm{cap}(x) = \max(-c_l, \min(c_u + \textrm{observedCost}, x)) cap (x)

= max (− c l ,

min (c u

+ observedCost ,

x)) withc l = 0.010

ETH c_l = 0.010 \;\textrm{ETH} c l

= 0.010

ETH andc u = 0.012

ETH c_u = 0.012 \;\textrm{ETH} c u

= 0.012

ETH , whereobservedCost \textrm{observedCost} observedCost is the gas cost of executing the settlement (which the solver pays).

Submitted scores that are non-positive will be ignored. If only one solution is submitted,referenceScore \textrm{referenceScore} referenceScore is set to zero. Formally, this corresponds to always considering the empty solution which does not settle any trades and has quality zero as part of the submitted solutions. Then one cannot win with a non-positive score and the reference score is at least zero.

Moreover, the score a solver reports is required to be smaller than the solution'ssuccessQuality \textrm{successQuality} successQuality , defined as theobservedQuality \textrm{observedQuality} observedQuality when the solution successfully executes. Since solvers are currently submitting the full solution along with their score, this is very easily checked; if the solution successfully settles on chain, itssuccessQuality \textrm{successQuality} successQuality is equal to the sum of the total surplus generated and the fees collected, and both are fully determined from the set of orders the solution intends to execute and the clearing price vector it reports.

note When including the gas costs of executing a transaction, the winning solver's payoff is bounded above byc u c_u c u . However, it is not bounded below because, in case of a revert, the solver pays at mostc l c_l c l to the protocol and may also incur gas costs. To limit currency mismatch, the solver receivesmin (payment , observedCost) \min(\textrm{payment}, \textrm{observedCost}) min (payment ,

observedCost) inETH andpayment – min (payment , observedCost) \textrm{payment} - \min(\textrm{payment}, \textrm{observedCost}) payment

– min (payment ,

observedCost) inCOW , using an up-to-date price. In other words, the solver payment is split into a gas reimbursement paid inETH and a reward paid inCOW . To arrive at the budget for consistency rewards, these per-auction rewards paid inCOW are deducted from the total rewards budget.

Specification of consistency rewards

Besides the per-auction rewards, solvers are rewarded for their consistency by distributing some additional COW to them proportionally to the number of auctions they provided valid solutions for. Specifically, suppose that the total per-auction rewards have been determined and expressed in COW and amount to some numberX X X of COW, as described above. Then, ifX < 250 , 000 X < 250,000 X

< 250 ,

000 , the budget that will be distributed as consistency rewards ismin { 250 , 000 – X COW , 6 ETH } \min{250,000 - X \textrm{ COW}, 6 \textrm{ ETH} } min { 250 ,

000

– X COW ,

6 ETH } , where again, to compare the two quantities, we use an up-to-date price.

Additional solver costs

Additionally, the winning solver might incur supplementary costs, such as, for example, negative slippage once a solution is settled on chain. These costs are not an explicit element of the mechanism, but they are relevant in determining the solver's optimal strategy. More precisely, perCIP-17 , solvers are responsible for managing potential slippage incurred by the settlements they settle. This is a component that affects payouts, but can be treated completely separately, and we do so inslippage accounting .

Solver bidding strategies

Apart from submitting their solutions, solvers must decide on a score to submit as a bid in the auction. In general, this score can be chosen freely by the solver (within the restriction0 < score < successQuality 0 < \textrm{score} < \textrm{successQuality} 0

< score

< successQuality). The recommended way of submitting a score is via reporting a success probability. The score is then computed as explained below.

Letp p p be the probability that a solution successfully executes, letsuccessCost \textrm{successCost} successCost be the costs that a solver pays if a solution successfully executes, and letfailCost \textrm{failCost} failCost be the costs that a solver pays if a solution does not execute successfully. For example, costs in case of success could include gas costs of the full settlement, while costs in case of failure could be small due to use of services like MEV Blocker.

Ignoring capping of rewards, the winning solver's expected payoff is

p · (successQuality – referenceScore – successCost) – (1 – p) · (referenceScore + failCost) . p \cdot (\textrm{successQuality} - \textrm{referenceScore} - \textrm{successCost}) - (1 - p) \cdot (\textrm{referenceScore} + \textrm{failCost}). p

· (successQuality

– referenceScore

– successCost)

– (1

– p)

· (referenceScore

+ failCost) . The optimal score is such that the solver wins if and only if it is profitable to do so, and is given by

optimalScore

p · (successQuality – successCost) – (1 – p) · failCost . \textrm{optimalScore} = p \cdot (\textrm{successQuality} - \textrm{successCost}) - (1 - p) \cdot \textrm{failCost}. optimalScore

= p

· (successQuality

– successCost)

– (1

– p)

· failCost . Accounting for the cap, the winning solver's expected payoff now is

p · (max (– c l , min (c u + observedCost , successQuality – referenceScore)) – successCost)

– (1 – p) · min (c l , referenceScore + failCost) . p \cdot (\max(-c_l, \min(c_u + \textrm{observedCost}, \textrm{successQuality} - \textrm{referenceScore})) - \textrm{successCost}) \ - (1 - p) \cdot \min(c_l, \textrm{referenceScore} + \textrm{failCost}). p

· (max (– c l ,

min (c u

+ observedCost ,

successQuality

– referenceScore))

– successCost)

– (1

– p)

· min (c l ,

referenceScore

+ failCost) . If there is no value ofreferenceScore \textrm{referenceScore} referenceScore such that the above expression is

strictly positive, then a solver does not want to participate. If, instead, there are values ofreferenceScore \textrm{referenceScore} referenceScore such that profits are strictly positive, then the logic discussed earlier continues to apply: each solver wants to win if and only ifreferenceScore \textrm{referenceScore} referenceScore is such that profits are strictly positive. Since the above expression is monotonic decreasing inreferenceScore \textrm{referenceScore} referenceScore , a solver should compute thereferenceScore \textrm{referenceScore} referenceScore such that the above expression is zero and submit that value as a score. Formally, the optimal score is now implicitly defined by

p · (max (– c l , min (c u + observedCost , successQuality – referenceScore)) – successCost)

– (1 – p) · min (c l , referenceScore + failCost) = 0. p \cdot (\max(-c_l, \min(c_u + \textrm{observedCost}, \textrm{successQuality} - \textrm{referenceScore})) - \textrm{successCost}) \ - (1 - p) \cdot \min(c_l, \textrm{referenceScore} + \textrm{failCost}) = 0. p

· (max (– c l ,

min (c u

+ observedCost ,

successQuality

– referenceScore))

– successCost)

– (1

– p)

· min (c l ,

referenceScore

+ failCost)

= 0. The above equation always has a solution that is independent ofreferenceScore \textrm{referenceScore} referenceScore . This solution is, in general, unique (that is, unique except for very specific parameters, in which case the solution is a closed interval).

Price estimation competition rewards (CIP-27)

The price estimation competition is a separate competition where solvers compete to provide the best response to a quote request. Quote requests look almost identical to single-order batch auctions, where there is only one order with a trivial limit price, and solvers propose executions of this order with the goal to maximize "out amount minus gas costs", in the case of a sell request, or minimize "in amount + gas costs" in the case of a buy request.

As specified inCIP-27 andCIP-36 , solvers that participate in the price estimation competition get rewarded as follows. Each order that is within market price, is associated with a quote that was computed as part of the price estimation competition, and was used in order to compute the limit price of the order. The protocol keeps track of the quote associated with each created order and the corresponding solver that provided the quote, and if and when the order gets executed, the solver that provided the quote (that can be different compared to the solver that ended up executing the order) gets rewarded withm i n 0.0006 E T H , 6 C O W min{0.0006 ETH, 6 COW} min 0.0006 ET H ,

6 CO W , where again, the conversion from ETH to COW is done by using an up-to-date priceEdit this page Previous Solver competition rules Next Slippage accounting