id-1.-deploying-the-carbonvortex-contract)

Using CarbonVortex

TheCarbonVortex is a contract designed to auction off fees collected within a Carbon DeFi ecosystem. It incentivizes arbitrageurs and public callers to interact with it and facilitates token trading through an auction mechanism.

This guide explains how to deploy and interact with the contract effectively.

1. Deploying the CarbonVortex Contract

Steps to Deploy:

1. Obtain the Source Code
2. Download the verifiedCarbonVortex.sol
3. from the officialBancor GitHub
4. repository.
5. Configure Deployment Parameters
6.
    - Token Address:
7.
    - Define the token for which fees will be auctioned.CarbonVortex
8.
    - supports 2 dimentional auction for increased flexibility:
9.
    -
        - targetToken
10.
    -
        - : This token will be the destination to which all fees collected in multiple different and unique tokens (stables such as USDC and USDT, popular such as wETH, wBTC, or meme such as PEPE, SHIB) will be auctions for. The result is that all tokens collected would be auctioned and traded for thistargetToken
11.
    -
        - eventually.
12.
    -
        - finalTargetToken
13.
    -
        - : This 2nd destination token allows the deployer to define another token in which thetargetToken
14.
    -
        - will be auctions for. This structure enables a secondary auction that can work for less liquid tokens yet allow consolidation. Deployer can decide to leave this address blank and avoid this step altogether.
15.
    - Auction Parameters:
16.
    - Allow flexibility in auction flow and speed as well as the process in which the auction price reacts to trades and restart the auction price as a result.
17.
    - Caller Incentive:
18.
    - Define the public incentive to encourage community engagement and interaction with the contract and auctions.
19. Deploy the Contract
20. Ensure deployment is on the intended blockchain network (e.g., Ethereum, Polygon).
21. Verify Deployment
22.
    - Check the deployed contract address.
23.
    - Verify the deployment on a block explorer to ensure all constructor arguments are set correctly.

1. Caller Incentive: When and Why to Trigger execute()

Theexecute() function plays a central role in the lifecycle of theCarbonVortex :

1. Transfers Collected Fees
2. : Aggregates fees from the protocol into the CarbonVortex contract.
3. Initiates an Auction
4. : Starts an auction to consolidate various tokens into thetargetToken
5. .
6. Provides Caller Incentives
7. : Rewards the caller of the function to encourage execution.

Caller Incentive:

- Caller Rewards:
- Callers are rewarded with a portion of the auction proceeds as defined in the contract.
- Profit Opportunity:
- Callers may profit by executing when they can acquire tokens below market value.

Steps to Identify Profitability and Use execute()

1. Evaluate Tokens Collected by the Contract
2.
    - Before callingexecute()
3.
    - , assess the amount of tokens currently held by theCarbonVortex
4.
    - contract. These tokens represent fees accumulated by the system.
5. Identify the Collected Tokens:
6. The contract incentivizes execution by offering rewards to the caller. This incentive is derived from the collected tokens that will be transferred and auctions when calling theexecute()
7. function.
8.
    - UseavailableTokens()
9.
    - to identify the available balance of the token at hand.
10.
    - Calculate caller incentives by multiplying this value by the returned value fromrewardsPPM()
11.
    - .
12. Trigger
13. execute()
14. Once the incentive is profitable, call theexecute()
15. function.

What Happens After execute() is Called?

1. Auction Initialization:
2.
    - The contract sets up a new auction or restart an existing one, with parameters such as:
3.
        - Starting price.
4.
        - Auction duration.
5.
        - Token amount for auction.
6.
    - Auction details are typically emitted as an event for external tracking.
7. Caller Reward:
8.
    - The caller receives their incentive, as calculated earlier, immediately upon execution.

1. Identifying Auction Prices and Trading Tokens

Fetching Auction Data

This process requires retrieving some information from the contract

1. Call theamountAvailableForTrading()
2. to identify the amount of tokens that are currently being auctions. This call requires passing the token address.
3. Call thetokenPrice()
4. to know the current price for the token
5. CalltradingEnabled()
6. andpairDisabled()
7. to verify the token auction is enabled
8. CallexpectedTradeReturn()
9. orexpectedTradeInput()
10. to confirm the required amount for the trade (this can also help in defining the minReturn correctly to avoid loses)
11. Executetrade()
12. to trade against the active auction in the contract

Alternatively, once can calculate the prices off-chain by analyzing the price dynamics:

- Auctions have a decaying price model half life (Dutch auction).
- As time progresses, prices decrease, incentivizing early bids.

Executing a Trade

Use thetrade() function to buy tokens:

- Specify the amount you wish to trade.
- Ensure sufficient token allowance or ETH (if applicable).
- Specify the maxInput (commonly known as minReturn) to avoid loses as a result of MEV activity.

Conclusion

TheCarbonVortex contract is a powerful tool for decentralized fee auctions. By understanding its mechanisms and incentives, teams can effectively deploy, maintain, and interact with it to maximize utility and ensure a seamless auction process. Previous Sample Subgraph Queries Next CarbonVortex Dynamics Last updated2 months ago