zAuction

OVERVIEW

An auctioning system for exchange of NFTs

GOALS

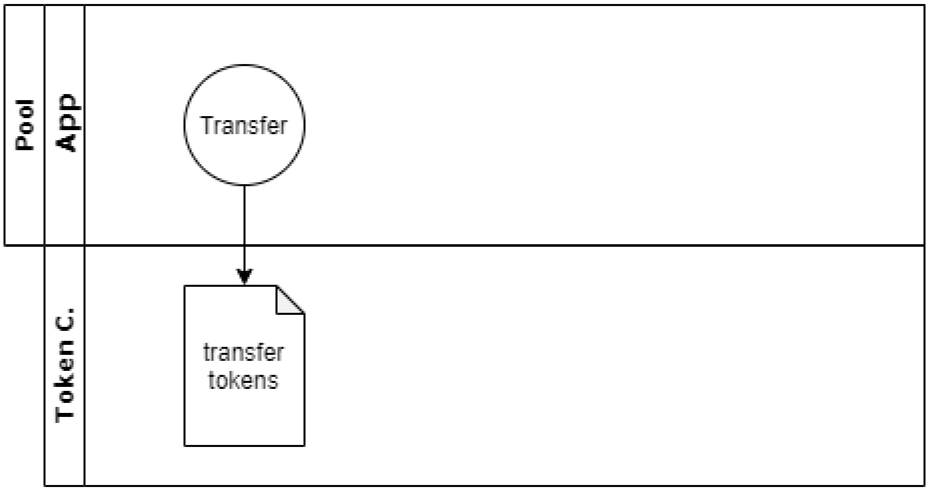
1. Buy/Sell ERC-721 and ERC-1155 NFTs
2. Auction NFTs
3. Upgradeable - Stateless functionality
4. Extendible - Add new auction types

SPECIFICATIONS

Users will be able to do simple transfer of NFTs.
Choose NFT -> Click Transfer -> Input ‘to’ address -> Confirm tx

erc1155 difference: also input token amount

This contract functionality exists in the token standards, so only needs ui to implement.

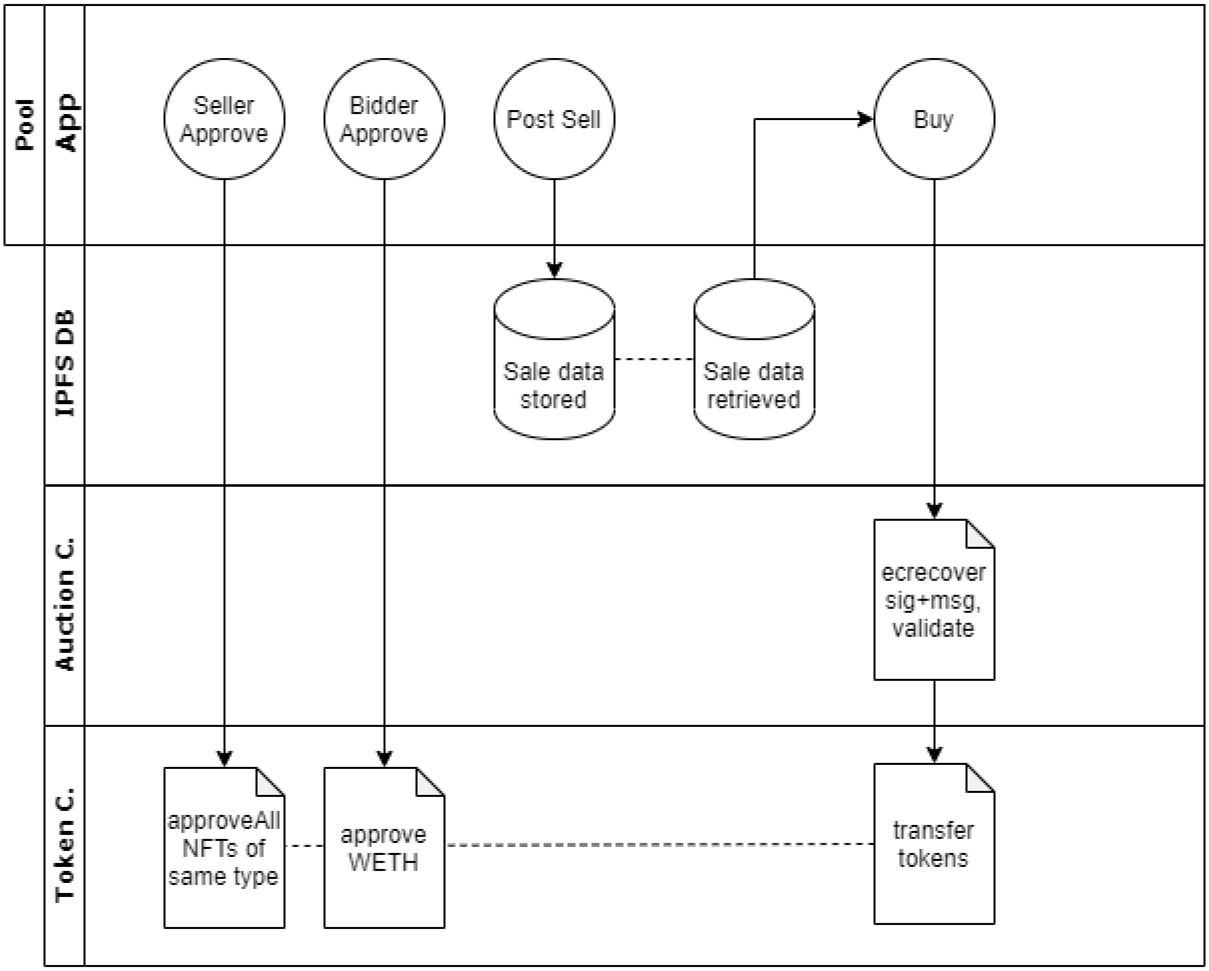


Users will be able to post NFTs at a sale price, and buy at that price.

Sell posts will be handled off-chain with signed transaction storage.
Seller: Choose NFT -> Click Sell -> Input eth amount -> Sign transaction -> Data & Tx hash stored w/ ipfs

Buyer: Choose NFT -> Click buy -> Seller's tx pulled from ipfs -> Confirm tx -> Seller's signature and sell data is sent with buyer's tx -> contract ecrecovers seller's signature and message, validates the hashed proposed sell data matches the signature's message hash, and transfers tokens

1155 difference: also input token amount to sell



Users will be able to post NFTs for auction, bid on auctions, and accept bids

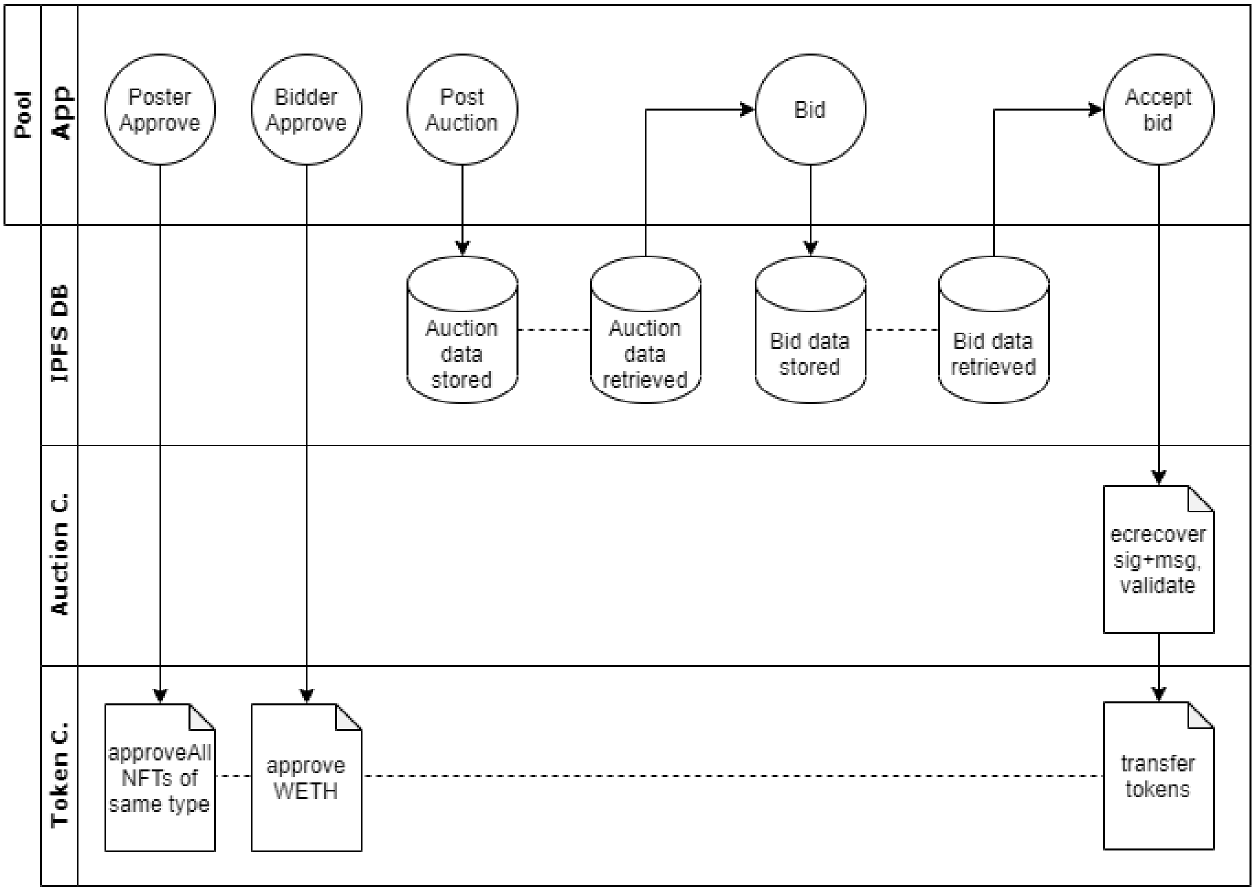
Auction posts will be handled off-chain with signed tx storage

Poster: Choose NFT -> Click Auction -> Store auction state in ipfs -> Approve auction contract for NFT

Bidder: Choose NFT -> Click bid -> Sign & store message -> Approve auction contract for WETH

-- 1155 difference: also input token amount to bid on

Poster: Accept bid: poster's message is pulled from ipfs -> confirm with bidder's signed tx -> poster sends accept tx with bidder's signature and the bet amount, NFT contract address, and token ID -> contract validates that the hash of the proposed buy data is equivalent to the signature's message hash, and transfers tokens



NOTES AND KEY RISKS

The above description doesn’t factor in the zAuctionAccountant that is used to manage eth balance state to allow for deposits and upgradeability.

an eth deposits, zAuction is stateless.Most of zAuction happens off-contract, where at the end of a post -> bid -> accept process, finalization happens in a single tx by the seller, who states “this is what the buyer said they’d pay for this NFT, and here’s their signature to prove it” (or reverse, buyer sends tx in simple sales)

So we just need to be sure that we are signing and validating messages securely and correctly. We need to be sure that we have eliminated rewind attacks on the signatures. .

To accomplish that we are hashing the zauction contract address, chain id, and a nonce with the messages, anything else needed?

The nonce system zauction uses is a global random nonce - a bidder submits a random uint256 with each new bid signature, the value is stored on acceptance and can’t be used again by anyone. This state is stored on zauction, not the accountant, and is lost on a zauction upgrade.

Any other types of attacks to be aware of here? We are using the openzeppelin ECDSA lib that takes care of the malleable signature considerations.

Any concerns with using IPFS to store signatures and bet data?