Hedgey OTC UAT Testing

Purpose: This document outlines all of the tests required to complete a successful OTC deal, for both current and futures deals, such that the code is sufficient for production usage. This does not constitute that there are no risks or bugs in the contract that would prevent hacks or unauthorized usage, just simply an outline of how the contracts are intended to work. Smart contracts are contained in this repository: https://github.com/hedgey-finance/OTC_Core

Overview:
There are two smart contracts, the HedgeyOTC and FuturesNFT contracts. The HedgeyOTC contract manages a peer to peer OTC exchange of tokens. A seller can create a new deal, stored as a struct, defining the parameters:

Seller: The person creating and selling the deal & tokens
Token: The tokens the seller is willing to sell
PaymentCurrency: The crypto currency the seller is willing to accept as payment in exchange for the tokens
RemainingAmount: The amount remaining to be bought
MinimumPurchase: The minimum chunk a buyer can buy, can be the entire amount or 0
Price: The price per token denominated in the payment currency
Maturity: The date which the deal expires and can no longer be purchased (ie time for the order to exist)
UnlockDate: The date when the tokens can be unlocked. Not a requirement, if the unlock date is < current time when a buyer buys tokens, then the tokens are immediately delivered to the buyer
Open: If the deal is open and can be purchased (ie Boolean to determine if it has been cancelled or not)
Buyer: An option to add a specific whitelist address for a specific buyer, so that only that person can but the tokens (or can be 0 address, implying that anyone can buy the tokens)

When a deal is created, the tokens are pulled from the seller address into the smart contract for escrow. Now this is an open deal. Buyers can now buy the tokens. When a Buy happens, a buyer pays to the seller the price per token * amount they want to buy (which must be greater than the minimum size). If the tokens are locked via a future unlockdate, then the tokens are transferred into the FuturesNFT contract, whereupon the contract mints an NFT representing the ownership of those tokens that can be redeemed at a future time after the unlockdate has passed. The buyer can redeem the NFT after the unlockdate and pull the tokens out of the contract. If the tokens are not locked (ie unlockdate is < current time) then the tokens purchased are immediately delivered to the buyer. The seller can at anytime cancel their order and withdraw the remaining tokens (which will be all of them if none have been purchased), whereupon no one can buy from that deal any longer.

Deployment: First the FuturesNFT contract is deployed (which is an Ownable contract). Then the HedgeyOTC contract is deployed, one of the constructor parameters is the address of the FuturesNFT contract. After the HedgeyOTC contract is deployed, the deployer and current owner of the FuturesNFT

contract will call the method to transferownership to the HedgeyOTC contract. This ensures that ONLY the HedgeyOTC contract can call special methods on the NFT contract, such as minting an NFT.

Deployment details (Rinkeby)

FuturesNFT Deployment:
https://rinkeby.etherscan.io/address/0xc6fd4f43e63edcc8cc31a178d57b3cd7206267d1

HedgeyOTC Deployment:
- 1% fee was added in the constructor arguments
(https://rinkeby.etherscan.io/address/0x1fea81bb1deff0263471f9c5f620e453729ac6fb

Test Wallets Used:

A	0x0C4FAb8d9DBE774708EeC313bf0295278E307bcD
B	0xde06fEee4c4e3A0B9eEA4f0Ed7a9b21F80a65C58
C	0xe31D847B47465cC2745319dAc9E0c6ac711cA10b
D	0x040B6bD961eEd76667D7b4F2a6615657C6b9a303
0	0x0000000000000000000000000000000000000000

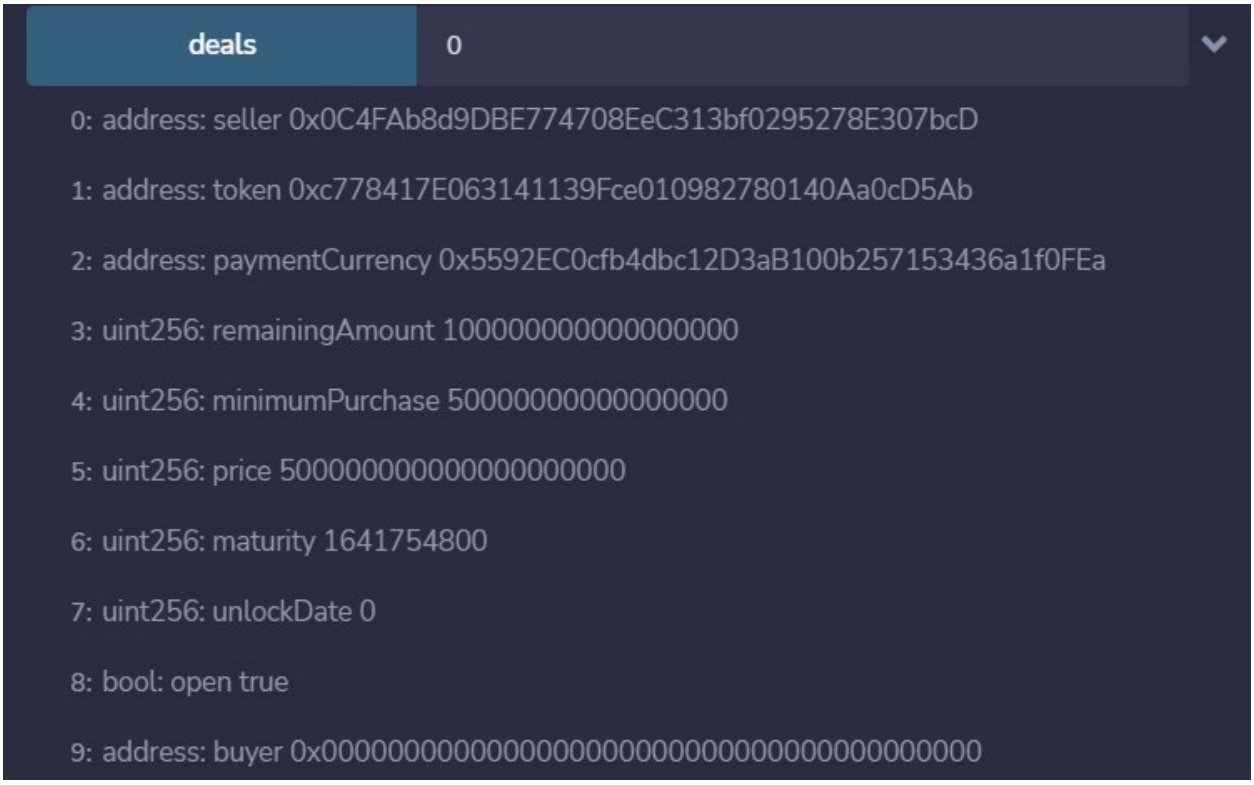
Tests to run on Hedey OTC:

1. **Create Deal**
   WalletA is going to create a new deal to sell .1 weth for DAI at a price of 500 dai / weth

   Parameters:
   a. Token: 0xc778417E063141139Fce010982780140Aa0cD5Ab (weth)
   b. paymentCurrency: DAI 0x5592ec0cfb4dbc12d3ab100b257153436a1f0fea
   c. Amount: 0.1 weth (1ee17)
   d. Min: 0.05 (5ee16)
   e. price: 500 (5ee20)
   f. maturity: 1641754800
   g. unlockDate: 0
   h. buyer: Address(0)

   Transaction:
   https://rinkeby.etherscan.io/tx/0x2f8ff31bc54a3625b265101a8cfa8ca6e589862014497e94343189f1254853e3

   Expected Results:
   A. Transfer 0.1 ETH from WalletA into the contract (wrapped as WETH)
   B. new Deal Struct created at index0 with the details above matching:

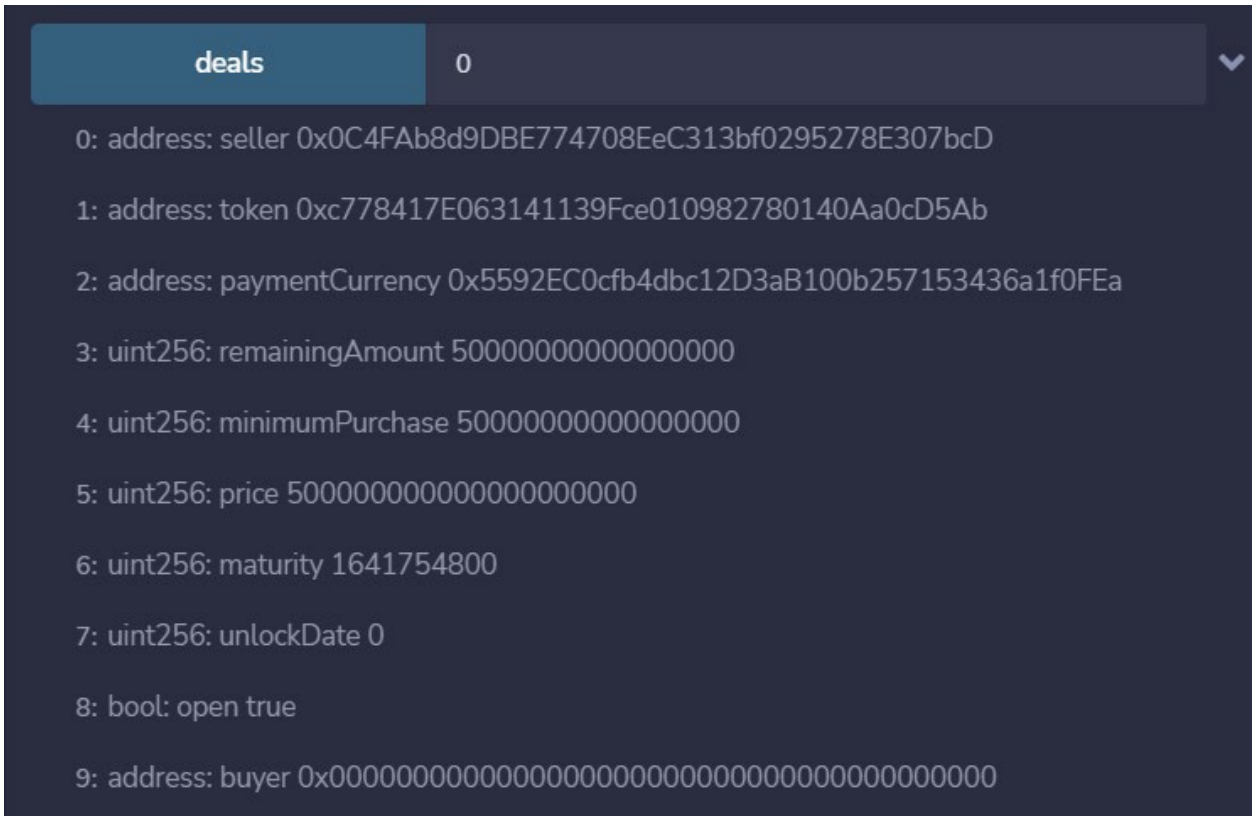


2. **Buy Partial Deal**
   WalletB will buy a partial amount of the deal, for 0.05 weth amount
   Parameters:
   a. Index (d): 0
   b. Amount: (5ee16) ie 0.05

   Transaction:
   https://rinkeby.etherscan.io/tx/0xa869733369a864f3b186fa1319e63556a4e5d455448eea23d7824d578d27aab1

   Expected Results:
   - WalletB transfers 25 DAI to purchase the tokens, and 1% sent to the fee collector (wallet) and reminder sent to WalletA.

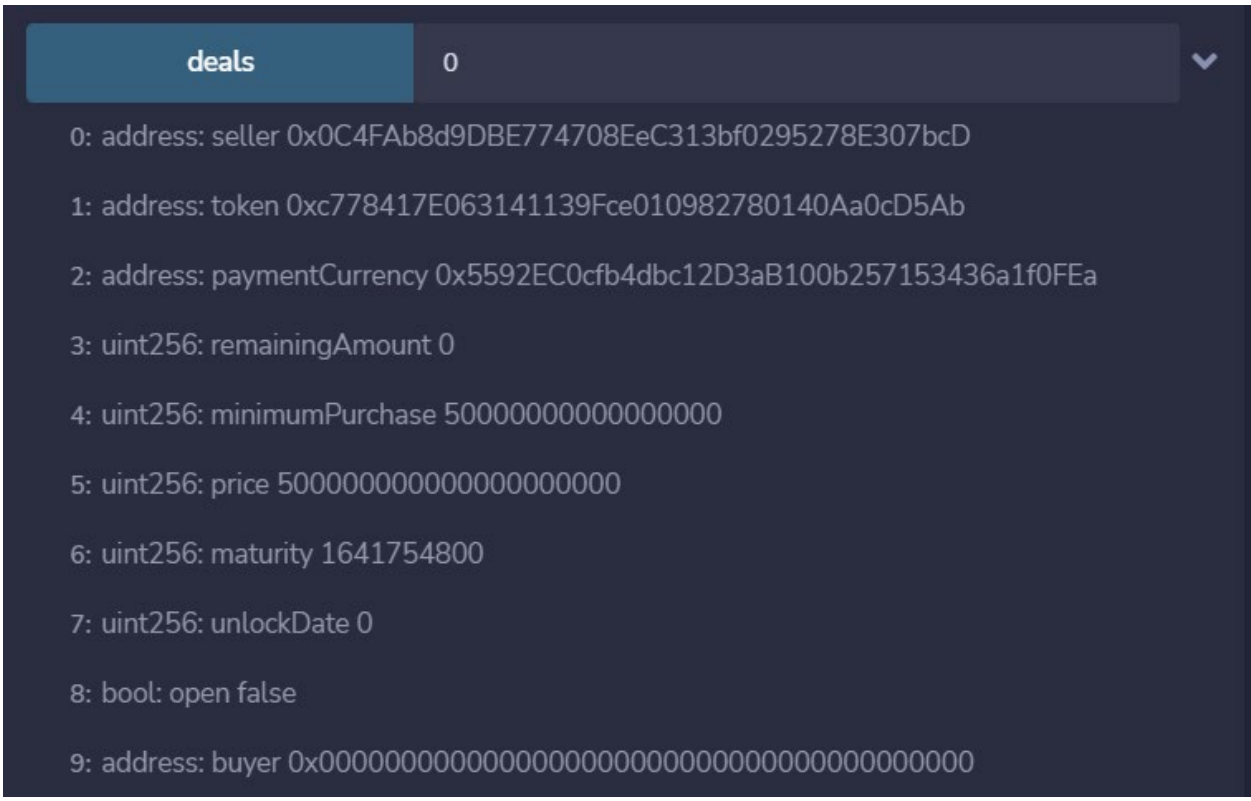
   - Because the unlock date was set to 0, the .05 ETH was delivered immediately to wallet.



3. **Cancel partially bought deal**
   WalletA will cancel the partially bought deal and have its remaining 0.05 ETH returned back to it
   a. Function: Close
      Index (_d): 0

   b. Transaction:
      https://rinkeby.etherscan.io/tx/0xa518cf295dd122f784ad3d0439d433903c9947c722eaaa22aa82fd245d007449

      Expected results:
      0.05 ETH delivered to accountA, and the deal struct is updated to reflect the Boolean open = false and remaining amount = 0



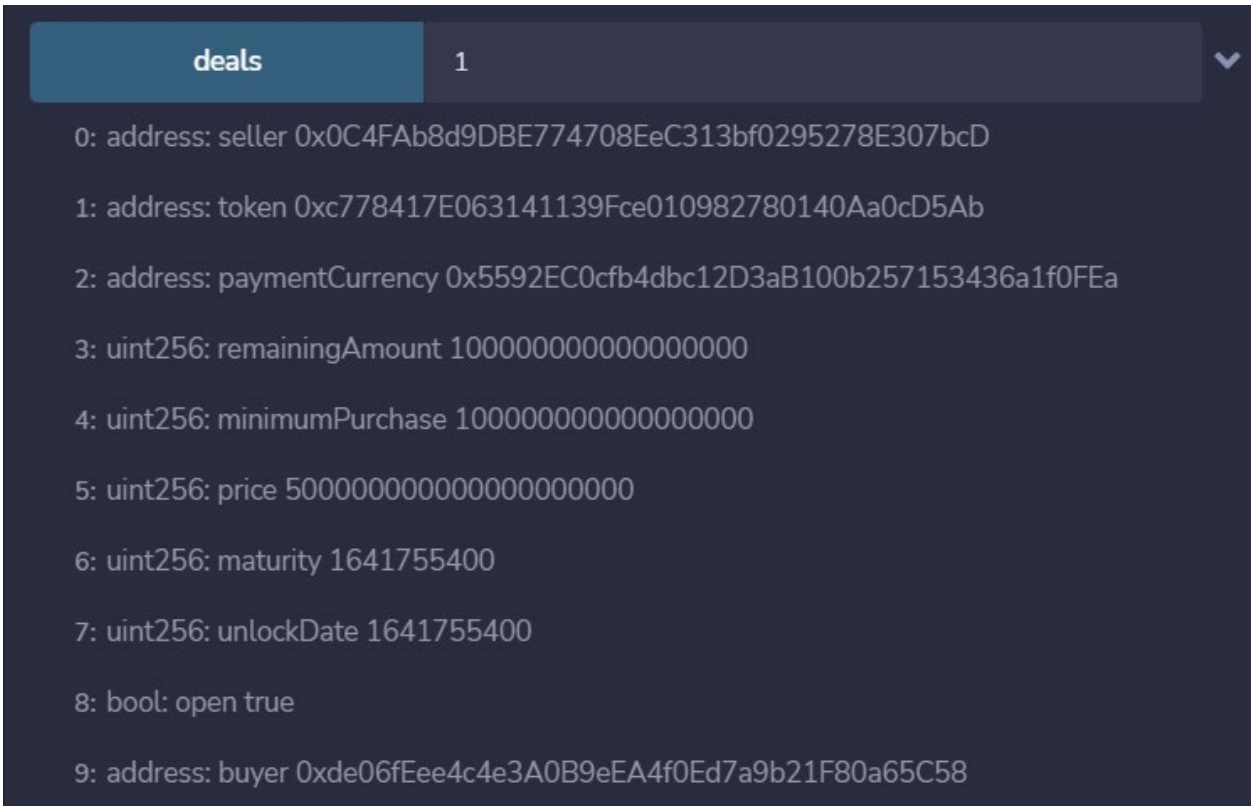
4. Buy the entire amount with dedicated buyer address
   WalletA will again make a similar deal with parameters as follows, but this time Only walletB can purchase the entire deal (and with a future unlock date)

   Parameters:

   a. Token: 0xc778417E063141139Fce010982780140Aa0cD5Ab (weth)
   b. paymentCurrency: DAI 0x5592ec0cfb4dbc12d3ab100b257153436a1f0fea
   c. Amount: 0.1 weth (1ee17)
   d. Min: 0.1 (ee17)
   e. price: 500 (5ee20)
   f. maturity: 1641755400
   g. unlockDate: 1641755400
   h. buyer: wallet (0xde06fEee4c4e3A0B9eEA4f0Ed7a9b21F80a65C58)

   Transaction:
   https://rinkeby.etherscan.io/tx/0x3565a610aabe633dd92f32e2596a0b5044cf21eaefb9f7aa0e35082a531a2191

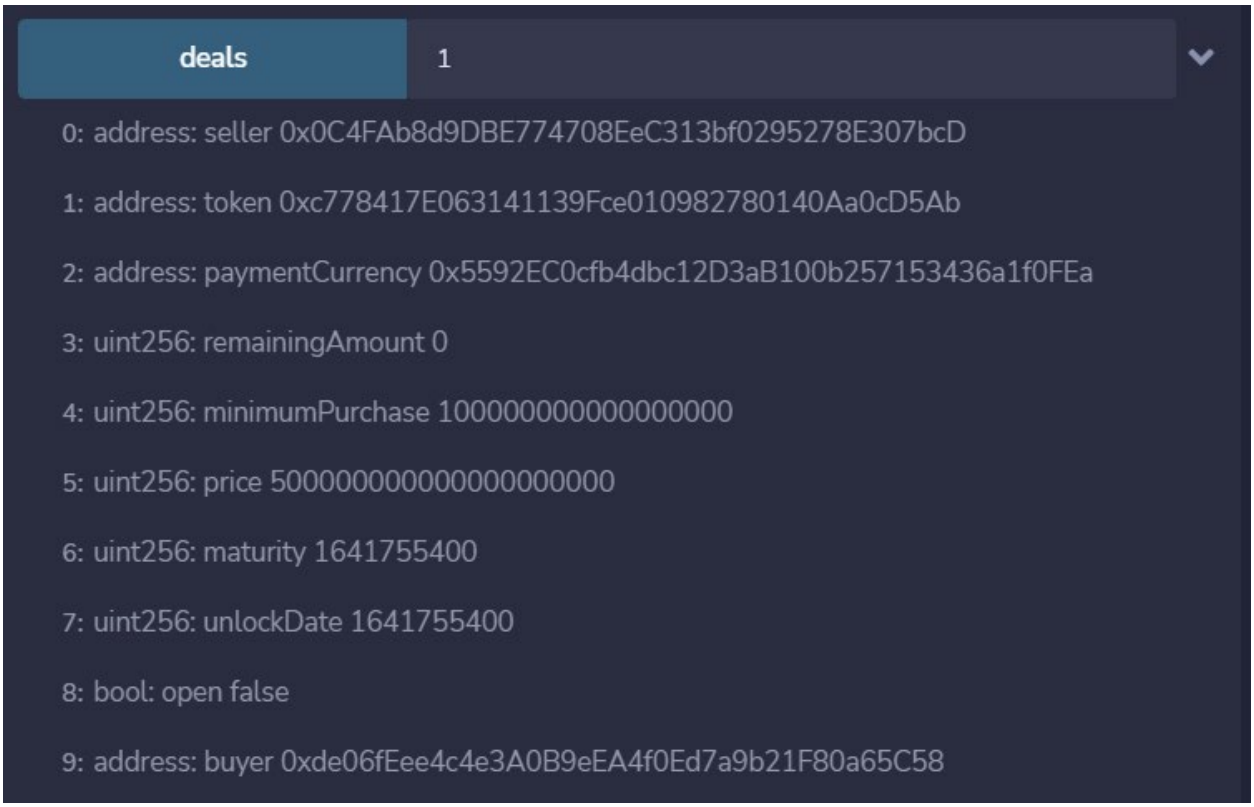
   Expected Results:
   Deal index1 should match the above parameters and be open = true



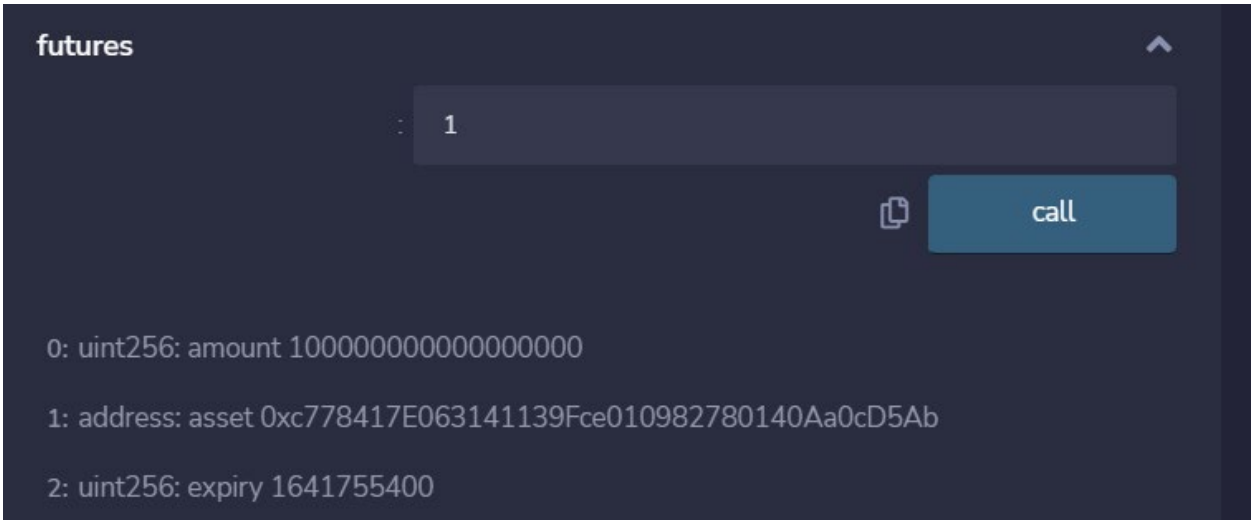
5. WalletB will purchase the entire amount of the deal
   function (buy)
   index (_d): 1
   amount: 1ee17
   Transaction:
   https://rinkeby.etherscan.io/tx/0x359c52ba3f836ef78370c928f25e87259a79910844f6c6215928b038cc86193c

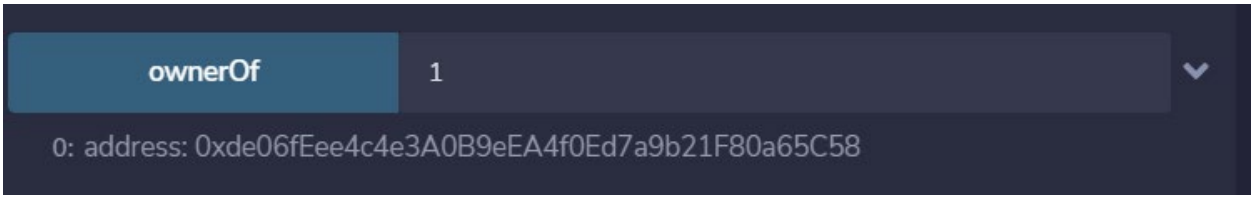
   Expected Results:
   WalletB delivers 50 DAI to pay for the .1 weth which is paid net of fees to walletA.
   Because this is a locked set of tokens, the 0.1 weth is transferred to the NFT Futures Address for escrow, and then an NFT (token id 1) is minted with WalletB as the owner of the NFT

   The OTC deal struct (index1) is updated to reflect that the remaining amount is now = 0 and the bool ‘open’ is set to false



On the NFT Futures contract, there is a struct created called ‘futures’, and at index 1 this struct should reflect the amount of tokens, the weth and the unlock date:





6. **WalletB redeems the NFT**
   WalletB owns a futures NFT at index 1, which entitles it the right to redeem 0.1 ETH. It will redeem that NFT, burn it and in exchange receive 0.1ETH from the contract.

   FuturesNFT Function: Redeem
   Parameter Index: _id = 1
   Transaction:
   https://rinkeby.etherscan.io/tx/0xa8f7a9c47728ec3172d23141e511b34486a78e5d0f3c279f92e2a4ca7c0cd4d2

   Expected Results: 0.1 ETH should be delieverd to walletB, and walletB will burn its NFT _id = 1. The struct should be updated such that there is nothing left to redeem and there is no owner for NFT _id = 1.