FiatDex Protocol

A decentralized method to trustlessly exchange Fiat currencies for cryptocurrency using non-custodial swaps and game theory.

Background: Currently, there are limited trustless method to exchange fiat currencies such as USD, EUR and others for cryptocurrencies such as Bitcoin, Ethereum or Litecoin. Most users seeking to convert fiat into crypto must use and trust a centralized gateway that serves as gatekeeper for entering and exiting crypto from fiat. Such gatekeepers may require information that users may not want to disclose such as Social Security Numbers, Driver's Licenses or Passports for fear that the information may not be held securely or will be abused. In addition, these centralized gateways can become honey pots waiting to be hacked. As seen with LocalBitcoins, there is a demand for a service that avoids the gatekeepers; however, as we have also seen with LocalBitcoins, such centralized services may become victims to burdensome regulations, whether it is right or not. There are decentralized exchanges that have fiat as an option, such as Bisq or LocalCryptos; however these types of exchanges use shared custody multisignature addresses with a trusted third party that must be the final arbitrator of all trades if the trading parties cannot resolve disputes themselves. Also, if the arbitrators disappear from the platform, the exchange can no longer function when there is a dispute.

What is FiatDex and how is it different? FiatDex is Ethereum based fiat to MakerDAO's Dai (DAI) swap smart contract. It can be ran as a backend protocol for any Dapp that is order matching users who want to swap fiat for crypto or vice-versa. It utilizes some features from various types of decentralized exchanges and puts it into a simple interactive smart contract. The Ethereum blockchain was chosen versus the Bitcoin blockchain as it has more flexible smart contracts and eventually users can swap their DAI for Bitcoin via other trustless methods if desired. DAI was chosen over native ETH due to its price stability. The DAI is designed to be pegged 1:1 with USD. FiatDex will be the decentralized gateway for users with fiat to enter the cryptoworld without having to trust a third party. The core tenet of FiatDex is **incentive**. Participants are incentivized to continue to the next step, and failure to do so in a timely manner will result in a penalty. Much like Bitcoin and cryptocurrency in general, incentive drives miners to mine and stakers to stake. Incentive will also drive FiatDex. Traders are not asked to implicitly trust each other but rather act out of their **own self-interest**.

How does FiatDex work: FiatDex is a contract with simple steps; however, there is a caveat regarding its use. **Both traders must already have DAI to initiate the swap**. Now let's start here:

Meet Alice and meet Bob. Alice has 200 DAI she wants to sell for $200 USD. Lucky for Alice, Bob has $200 USD that he wants to use to buy 200 DAI.

1) Prior to the trade, Alice and Bob communicated that they want to exchange and how to do the exchange, including sharing a common hash between each other that will be used as a trade ID. This communication can be Telegram, IRC, email or anything else.

2) Alice creates a swap position using the trade ID in the smart contract with a value of 200 DAI, this swap position requires a 150% collateral (in this case 300 DAI), so Alice must send 500 DAI in total.

3) Bob sees that Alice has opened the swap position with the correct amount and then sends his collateral to the same swap position which is the same at 150% of the sent amount, so Bob must send 300 DAI.

4) The swap position now contains 800 DAI.

5) After sending his collateral, Bob must now send $200 USD to Alice. The preferred method is some sort of irreversible cash transfer (not bank to bank as those tend to be reversible).

6) When Alice receives the money and verifies it is not fake, she will then close the swap position unilaterally. This automatically triggers payment from the smart contract to Bob for 200 DAI as well as returning the collaterals to each respective party. Alice will get her 300 DAI collateral back and Bob will get his 300 DAI back. Both parties are happy and Bob is 200 DAI richer (though $200 poorer) than when he started. Alice is $200 richer (though 200 DAI poorer) as well. There is a 1% service fee on the swap so Bob is actually 198 DAI richer.

7) If any party misbehaves during the swap, either party can take unilateral action to punish the other party, which will also punish themself.

Sounds simple, but what happens if...

- Alice doesn't open the swap position

  - Then Bob doesn't send his part of the collateral as he can see that there is nothing in the swap position on the blockchain. No one loses DAI or fiat.

- Alice opens the swap position and Bob hasn't put any DAI into it

  - Alice can refund from the swap position without penalty as long as Bob has not put his collateral into the swap position. For refunds, no fee is charged to Alice.

- Bob doesn't send the cash or cash is not real / not enough

  - Alice will not finalize the swap position until she is satisfied. If she never closes the swap position, Bob can abort the swap position and charge a collateral penalty between 2.5-100%. **This penalty is deducted from both Alice's and Bob's collateral equally** so it is in their collective interest to complete the swap as planned and not utilize the penalty.

- Alice receives all the money, but declines to close the swap position

  - Until Alice finalizes or aborts the swap, she will not receive her collateral back, so whatever gain she has made will be less than the collateral. In addition, Bob can abort the swap and penalize Alice (and himself) if the swap is taking too long.

- Alice drags her feet when finalizing the swap position or Bob takes too long to (or doesn't) send the cash after putting in his collateral

  - Both Alice and Bob can unilaterally abort the swap if the trade is proceeding unsatisfactory, although the collateral penalties are different depending on who aborts. When Alice aborts, she can chose a penalty between 75-100% of the collateral amount, and when Bob aborts, he can chose a penalty between 2.5-100% of the collateral amount. Collateral penalties affect **both parties equally** so if Bob aborts and chooses a 50% penalty, he will receive 50% of his collateral back (150 DAI) and Alice will receive 50% of her collateral back (150 DAI). In any case, Alice will receive her sending amount back (200 DAI). The penalty goes towards the contract owner to be used towards maintenance for the platform.

FiatDex Gateway: FiatDex Gateway is a simple client based web interface to interact with the FiatDex protocol. It requires MetaMask to use and is a useful starting point if you plan to integrate FiatDex protocol into your own Dapp. The website is best ran on a localhost server as MetaMask requires a server connection.

FiatDex Marketplace: FiatDex Marketplace is a simple display that shows buy and sell orders for people interested in using the FiatDex protocol. Users can anonymously post requests that are temporarily stored on a central server.

FiatDex Protocol is usable now

Visit etherscan to understand the contract functions and to implement the contract into your own Dapps.

ETH Mainnet Address: 0x75a2a05b8a21568f5e052b6bbcfb799624fb2d8e

External Contract Functions

openSwap(bytes32 _tradeID, address _fiatTrader, uint256 _erc20Value)

This function opens the initial swap position. Alice is the daiTrader who calls this function to create the swap. The tradeID is a variable that is shared between daiTrader (Alice) and fiatTrader (Bob). The DAI that is sent to this function is unevenly split into amount being sent to fiatTrader and amount being held as daiTrader collateral.

addFiatTraderCollateral(bytes32 _tradeID, uint256 _erc20Value)

The fiatTrader sends DAI to this function to add collateral to the swap. Only the fiatTrader can add DAI into the swap position with this function. The DAI required must be equal to or greater than the collateral already present from daiTrader otherwise the transaction will fail. Once the fiatTrader has done this and confirmed a successful transaction, the fiatTrader needs to now send fiat to the daiTrader.

refundSwap(bytes32 _tradeID)

This function will refund the DAI from the daiTrader if the fiatTrader hasn't already put in his collateral. The daiTrader cannot refund if the fiatTrader has already put in collateral into the swap position. No fee is charged for a refund.

finalizeSwap(bytes32 _tradeID)

This function will complete the swap position. The daiTrader cannot complete a swap position unless the fiatTrader has put funds into the collateral. Only the daiTrader can close the swap. The collateral is returned to each trader and the fiatTrader will get the sent amount from the daiTrader minus the 1% service fee. The daiTrader should only close the swap if she has **confirmed and verified** the fiat she has received from the fiatTrader.

fiatTraderAbort(bytes32 _tradeID, uint256 penaltyPercent)

This function will abort the swap position. It can only be called by the fiatTrader. The fiatTrader may have various reasons why to abort the swap, such as the inability to send fiat to the daiTrader. There is a penalty required to pay when aborting that will be deducted from both traders collateral; however the sending amount will be returned in full to the daiTrader. The penaltyPercent in this case is an integer that needs to be between 2500 and 100000 which represents 2.5% and 100% internally.

diaTraderAbort(bytes32 _tradeID, uint256 penaltyPercent)

This function will abort the swap position. It can only be called by the daiTrader. The daiTrader may have various reasons why to abort the swap, such as if the fiatTrader fails to send the correct amount of fiat or fake fiat. There is a penalty required to pay when aborting that will be deducted from both traders collateral; however the sending amount will be returned in full to the daiTrader. The penaltyPercent in this case is an integer that needs to be between 75000 and 100000 which represents 75% and 100% internally.

changeContractOwner(address _newOwner)

This function is called only by the current contract owner and it will change ownership to the new owner. The only role of the contract owner is to collect fees and penalties. The owner cannot destroy the contract or stop the swaps. The first contract owner is created when the contract is created and it will be the address that deployed the contract.

External Viewable Contract Functions

viewSwap(bytes32 _tradeID) returns:

(uint256 swapState, uint256 sendAmount, address daiTrader, address fiatTrader, uint256 openTime, uint256 daiTraderCollateral, uint256 fiatTraderCollateral, uint256 feeAmount)

This function is an ETH Call function that provides information about the state of the swap position to both the daiTrader and the fiatTrader. They will use this information to determine where they are in the trade and what they should do. Possible swaps states are: NOTOPEN (0), INITIALIZED (1), REFUNDED (2), ACTIVE (3), CLOSED (4), FIATABORT (5), DAIABORT (6). NOTOPEN is the default state prior to when the daiTrader opens the swap position. INITIALIZED is the state when the daiTrader opens the swap position, waiting for the fiatTrader to put in the collateral. REFUNDED is the state when the daiTrader has received a refund from the swap position prior to fiatTrader putting in collateral. ACTIVE is the state when the fiatTrader has added the correct collateral. At this point the fiatTrader needs to send the fiat. CLOSED is the state after the daiTrader has finalized the swap position successfully. FIATABORT is the state after the fiatTrader has canceled the swap position and returned the collaterals. DAIABORT is the state after the daiTrader has canceled the swap position and returned the collaterals.

viewFiatDexSpecs() returns:

(uint256 version, address owner)

This function is an ETH Call function that provides information about the contract in general. version is the version of the contract and owner is the address that currently owns the contract and collects the fees.

Date: Thursday, June 8th, 2020