bZx

Margin. Decentralized.

Whitepaper

Abstract

bZx is built on Ethereum and integrated with the 0x protocol. It is the first fully decentralized, peer-to-peer margin funding and trading protocol.

bZx is not itself an exchange, but a protocol that can be integrated into the current exchange infrastructure. Exchanges and relays are incentivized by fees denominated in the BZRX protocol token (BZRX) to offer decentralized margin lending and margin trading services. Assets are valued and liquidated via competing oracle providers. By decoupling the valuation and liquidation of assets from the protocol, the oracle marketplace approach allows competition to drive the oracle provider fee to its marginal cost while encouraging experimentation and flexibility.

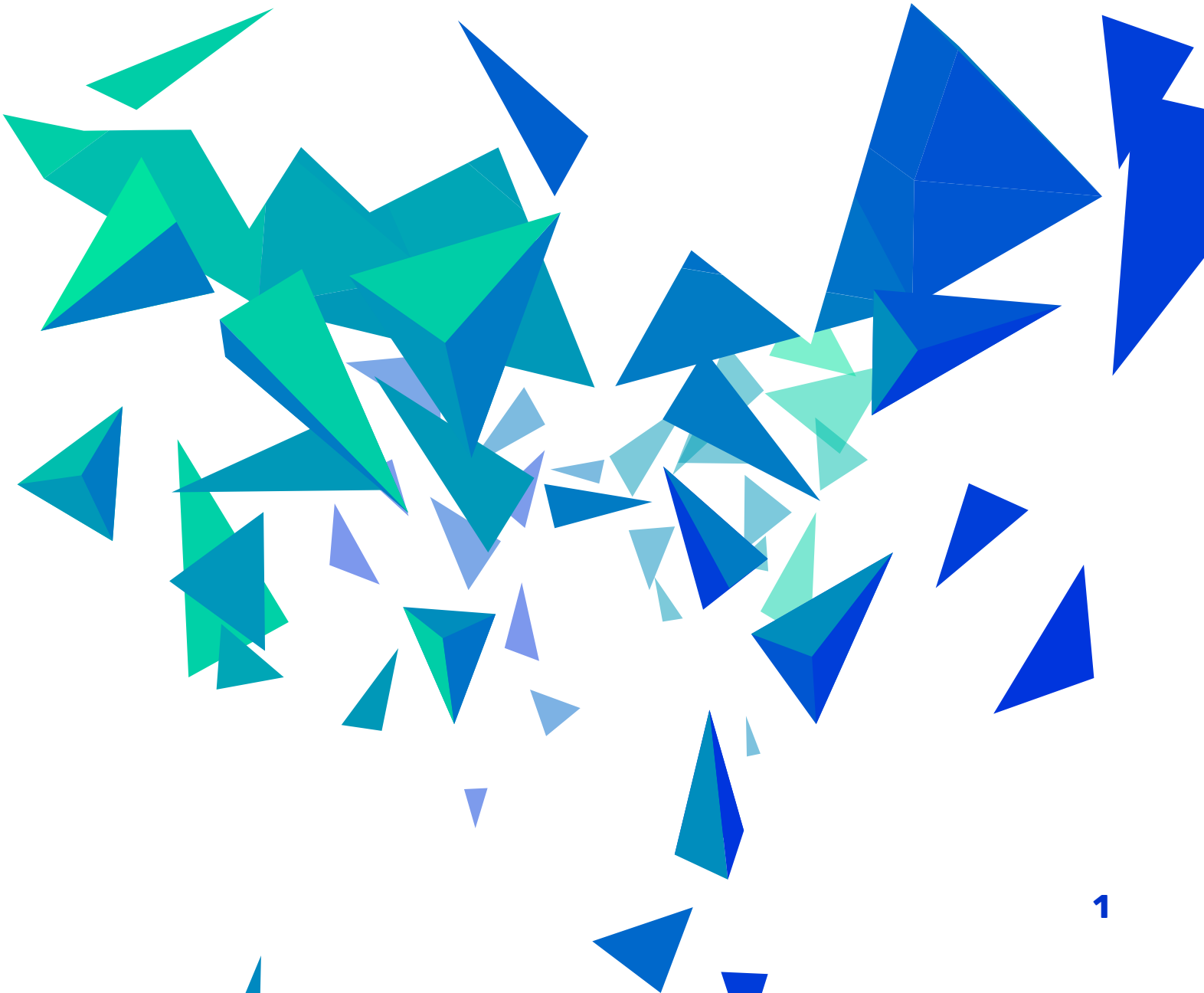


Table of Contents

PHASE 1

Motivation

One of the persistent contradictions of the cryptocurrency space has been the theme of decentralized assets traded on centralized exchanges.

In the wake of the 0x revolution, a new generation of decentralized exchanges (DEXs) are taking root. These decentralized exchanges address some of the existing problems with older DEXs, while still lacking the capabilities of many of the leading centralized exchanges. Individuals looking to engage in margin lending or margin trading are still forced to funnel their liquidity to centralized token and coin exchanges, exposing them to an additional form of counterparty risk.

Counterparty risk is encountered when the risk of a third party defaulting jeopardizes the assets of an investor. Margin lending exposes the lender to counterparty risk both from the exchanges and the borrower. The specific type of avoidable counterparty risk incurred by lenders and borrowers using centralized exchanges is called *custodial risk*; allowing individuals to maintain control of the private keys to their wallets at all times obviates this risk. Lenders face additional counterparty risk from underwater borrowers who fail to be liquidated in time.

Decentralized margin comes with significant technical challenges. The most significant challenge is the design of a reliable oracle that can match the settlement security of centralized exchanges. In the context of margin lending, the oracle problem is caused because Ethereum contracts are not natively aware of asset prices on or off the blockchain. If smart contracts can't stay aware of asset prices on the open market, they can't consistently force-liquidate borrowers on that market to protect lenders from adverse movements. The most serious obstacle to decentralized margin lending is being able to reliably and securely liquidate troubled positions. The bZx protocol serves as an on-chain solution to these challenges.

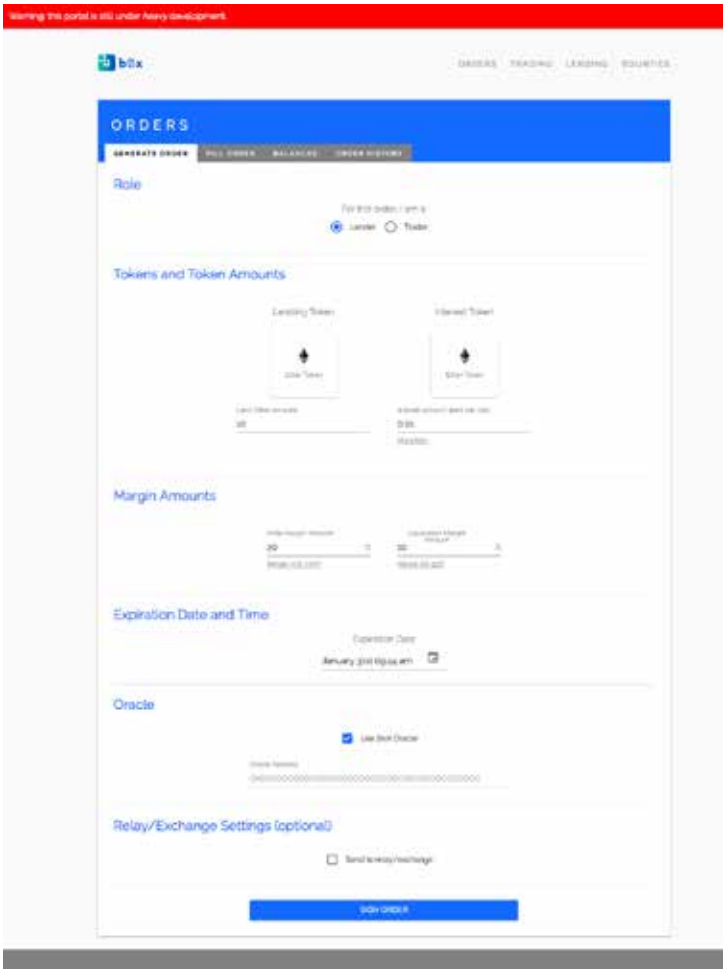
The Components of bZx

bZx.js Library

The bZx.js library is a promise-based asynchronous JavaScript library that contains all the functions needed to interact with bZx smart contracts on-chain. Software developers can use this library to easily integrate with and develop for the bZx protocol. Relays and exchanges will use this library to build an interface for margin lending and trading on bZx, providing a value-add to their customers. These relays will be able to add a funding tab, similar to many centralized exchanges. In the same way that 0x.js allowed relays to easily create a frontend for exchanges, bZx.js will do likewise for funding.

bZx Portal

The bZx Portal is a web-based decentralized application that serves as a frontend to the bZx protocol, utilizes the bZx.js library, and serves as a one-stop shop for individuals looking to interact with the protocol for margin lending and trading. There is no requirement to use the bZx Portal for lending or trading on bZx, but it provides a convenient access point for users that aren't otherwise on an exchange or relay.



The initial release of the bZx Portal will be split into four sections:

1. A section for the lender and trader to make or take bZx loan orders.

2. A section for the trader to manage the loan once the funds are lent, including the opening of trades, the closing of trades, and ending a loan early.

3. A section for the lender to manage the loan once the funds are lent including reviewing how their funds are being used and requesting an interest payout.

4. A section for bounty hunters to manage open trades for margin liquidation, and to liquidate if needed.

bZx Smart Contracts

The bZx protocol is a series of smart contracts that facilitate on-chain margin lending, and the opening, monitoring, and liquidation of ERC20 token trades. The entry point for all state changing transactions with bZx is the *bZx.sol* smart contract. This acts as a controller of other sub-contracts that make up the various parts of the protocol, including *bZxVault.sol*, *bZxTo0x.sol*, and the custom Oracle contracts for trade management (discussed in Section 2.3).

- ***bZxVault.sol:*** An escrow contract for storing ether and tokens not involved in active trades.
- ***bZxTo0x.sol:*** An interface for taking trades using the 0x Exchange contract. The contract can be easily upgraded later if ZeroEx makes breaking changes to their on-chain exchange.
- ***Oracle_Interface.sol:*** An interface provided as a starting point for developing custom Oracle contracts that interact with the bZx protocol and make up the oracle marketplace. An inheriting contract must implement all provided function declarations to work properly with bZx. Though funds are escrowed in the bZxVault, trades are escrowed by the oracle, meaning the oracle and not the bZx protocol has sole discretion to withdraw or liquidate the funds within the constraints of the protocol logic.

Care was taken in contract design to allow for seamless future upgrades with minimal disruption, and protection of user funds. The decentralized governance aspect of upgrade management will be discussed later.

The interaction with the bZx smart contracts begins when a lending order is sent to bZx.sol. Whether a person wishes to lend funds or borrow funds, this lending order is generated when that person defines the loan parameters on a 3rd party relay (integrated with bZx.js) or on the bZx Portal directly. This order is broadcast through any medium, though most likely through a relay. Takers bring signed orders to the bZx contract, initiating the margin lending process. A series of competing oracle contracts form the basis of an oracle marketplace and can be selected by users based on their preference, as part of the order creation process. The oracle contract chosen is responsible for overseeing positions taken using that oracle, managing the price feed, and controlling liquidation logic. Any desired oracle can be chosen by the user when the order is created, as long as the oracle has been registered in the bZx Oracle Registry contract. Oracle contracts that make it into this registry will have their source code published and will have been publicly vetted and accepted through decentralized governance, ensuring they are safe and reliable. bZx intends to maintain a marketplace of these oracles, allowing for public rating, and providing a means to select an oracle that meets the public's qualifications.

The Oracle

Design Decisions

Oracles introduce a significant complication to the decentralized margin lending problem. Even if an oracle is decentralized and interfaces with on-chain price information, network congestion can pose a significant threat to its ability to liquidate a position in a timely manner. The approach bZx has chosen is to create an oracle marketplace where oracle providers compete, allowing providers and users to select the trade-offs tailored to the individual. This system will allow oracle services to be provided at the lowest possible fees with the highest reliability. Any individual or organization will be able to create their own oracle. If the oracle is successful, the creators will either profit from token schemes facilitating seigniorage or from fees on interest earned by the lender. An oracle provider can charge any fee they want, but individuals may choose not to use them if the fee is too high.

Our own flagship open-source oracle is presented here: bZxOracle.
bZxOracle is fully decentralized and operates partially off-chain. When creating an order, an oracle provider must be specified. If the bZxOracle is specified, then anyone can call the liquidateTrade contract method and receive a bounty when the proper conditions are met. Support is provided at the protocol level to allow third-party oracles to impose whatever constraints necessary on when the liquidateTrade method can be called. With bZxOracle, bounty hunters keep track of all open trades taken using bZx, determining whether any have gone below margin maintenance. This pushes the most computationally intensive tasks off-chain. When a bounty hunter has determined that a position has gone below margin maintenance, they call into bZx to liquidate the trade with bZxOracle.

After the liquidateTrade method has been called, the *bZx* contract makes a call to the *bZxOracle* contract *(fig.1)* to determine whether the position has gone under margin maintenance. The *bZxOracle* contract pulls from the most liquid three decentralized exchange APIs. The average disagreement between each price provided by the API is calculated. The DEX which provided the number with the highest average disagreement is discarded and the two remaining DEXs are used in the calculation of the volume weighted average price. This is done to prevent temporary, erroneous prices from allowing a borrower to be wrongly liquidated. This also provides protection against bounty hunters seeking to maliciously liquidate orders to extract a greater sum of bounties. While more sophisticated outlier detection algorithms might seem preferable, we have opted for this method to limit gas costs. We will initially only be using the on-chain price feed from KyberNetwork until other secure on-chain price feeds come online.

When the liquidateTrade method is called for a trader's open position, and the trade is confirmed to require liquidation, an on-chain trade is made to close the position. During times of high congestion, it becomes increasingly likely that any single transaction sent will be knocked out of the mempool and fail to be mined. Traders are familiar with this phenomenon during popular ICOs, where network congestion has prevented many individuals from successfully participating in a token sale.

One of the principal design considerations for bZxOracle was to have it work in even extraordinary circumstances. Once an account is identified as being below margin maintenance, it is imperative that it is liquidated as quickly as possible with the fewest number of transactions required to be mined. This is why we crowdsource liquidation calls. This is also why we have designed the system to work without user intervention even during times of market crisis.

We strive to make the contract as economical as possible with regard to gas usage. We are investigating the use of TrueBit to off-load DEX API calls and liquidation gatekeeping calculations off-chain. This is the architecture behind v2.0 of bZxOracle and continues to be an area of active research.

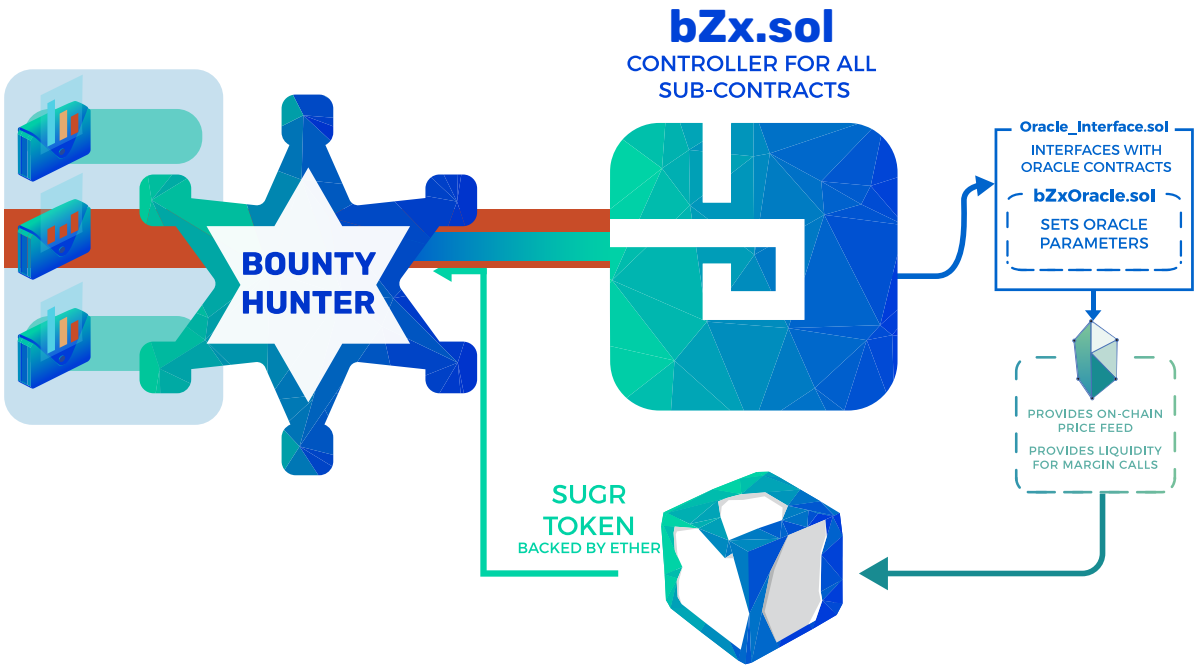


Figure 1. Simplification of the liquidateTrade method.

Oracle Governance

In bZxOracle, a ten percent fee will be collected from the interest earned by lenders and will be used for several functions, including: decentralized governance, bounty hunter incentivization, gas fee refunds, and systemic risk insurance. By contrast, many centralized exchanges charge fifteen percent or more of the interest earned. Since bZxOracle will be competing in an oracle marketplace, market forces will eventually force the oracle fee down to the marginal cost of providing the service. We encourage individuals to fork the open-source code and compete with their own variant of our oracle if they believe they can deliver better results. For example, in times of low network congestion, a lender might feel comfortable using a fee-free fork of the code and being their own bounty hunter. While this forgoes the protection of a network of individuals crowd-sourcing transactions, this trade-off might be preferable for some lenders and borrowers.

Bounty hunters will be paid a bounty to compensate them for the gas spent calling the contract and the resources spent monitoring margin account health off-chain. In order to establish the average price of gas on the network, we extract the gas price data from the takers bringing signed orders to the *bZx* contract. We use this data to update an exponential moving average (EMA) which represents the price of gas on the network. The higher the gas price used in prior transactions, the higher the bounty will be. This will enable the bounty to dynamically scale in sync with network congestion, ensuring that liquidation transactions always maintain priority in the transaction queue. If this bounty proves insufficient or excessive, it can be modified by the oracle's decentralized governance mechanism discussed below.

There are two trade-offs we considered when setting the bounty size:
The first is that if the bounty is set too high, a race condition will cause bounty hunters to use a gas price far above what is necessary to be at the front of the transaction queue, burning gas to compete with other bounty hunters. This would create great deadweight loss, compensating miners far more than is necessary to process the transaction quickly. The second trade-off is that if the bounty is set too low, the average gas price used to process the transaction won't be enough to make it past the transaction queue, causing liquidations to take too long or not be mined at all. After determining an upper bound on gas used by calls to the liquidateTrade method, the bounty will be set such that bounty hunters are incentivized to send transactions with a gas price slightly above the average.

The fees collected by bZxOracle will be tokenized and distributed to its users *(fig. 2)*. Lenders and borrowers will receive Sugar (SUGR) tokens to compensate them for taker fees and gas. Bounty hunters will receive Sugar tokens as their bounty. This token will be used to decentralize governance of the oracle, giving the individuals invested in the network a vote in proportion to their usage. SUGR tokens are backed by Ether and redeemable for a fixed percent of the bZxOracle reserve. As bZxOracle collects more fees, the Ether reserve grows along with the value of the SUGR token. The distribution of the SUGR token will take strong inspiration from Ethfinex and their Nectar token. The SUGR token roll-out will be incremental, with the oracle initially distributing Ether to the lenders, borrowers, and bounty hunters.

The volatility of cryptocurrencies create a high risk that a cascade of margin calls could cause a flash crash. Part of the fees collected by bZxOracle will be set aside into a decentralized insurance fund denominated in Ether and BZRX token to protect lenders in the event of a black swan event. If many lenders have lost principal due to abnormal market circumstances, the holders of the Sugar token are incentivized to use the insurance fund to restore the principal of the lenders. This maintains the reputation of the network as a safe place to loan funds and safeguards the future revenue of the Sugar token. Part of the task of governance of the oracle is to decide what loan order parameters are insurable. It might be decided that providing loans for margin traders using leverage over a certain threshold are ineligible for the oracle's insurance.

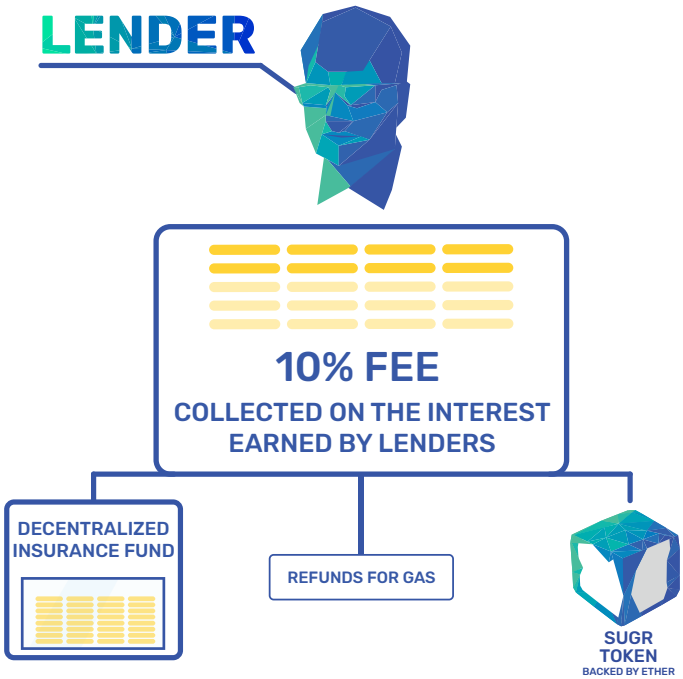


Figure 2. Visualization of collected fee distribution.

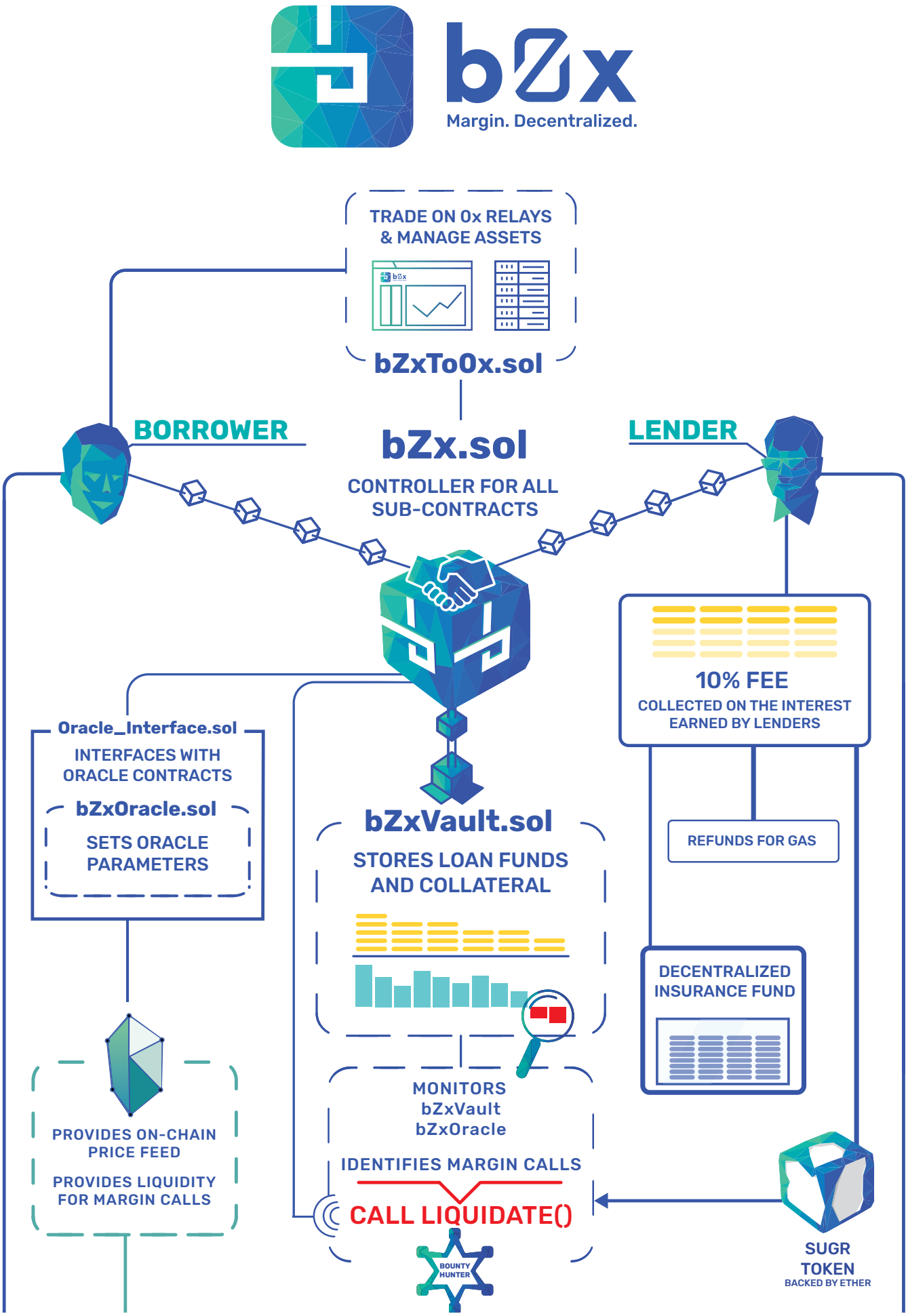


Figure 3. Infographic of bZx protocol.

SPECIFICATION

The BZRX Token

The BZRX token is a utility token with two main functions:

1. The incentivization of order book aggregation by relays
2. Governance of the bZx protocol

The BZRX token functions for the bZx protocol much like the ZRX token functions for the 0x protocol. Both tokens coordinate networks of rational economic agents around a protocol. Both are used to facilitate continuous, decentralized updates to the protocol.

Token Governance

The most common directions for governance in the space are Aragon and multi-signature wallet arrangements leading to a DAO. It has become clear that multi-signature wallets represent an unnecessary attack surface. On this issue we will continue to solicit feedback from the community. We are currently investigating the use of an upgradeable DAO.
A Medium post detailing our plans for governance will be released prior to the crowdsale.

Broadcast Orders

Name	Data Type	Description
bZxAddress	address	Address of the bZx smart contract.
makerAddress	address	Address of the maker (lender or trader) of the lending order.
loanTokenAddress	address	Address of the ERC20 token contract to be loaned to the trader.
interestTokenAddress	address	Address of the ERC20 token contract to be paid as interest to the lender.
collateralTokenAddress	address	Address of the ERC20 token that the trader put up as collateral.
feeRecipientAddress	address	Address of an exchange or relay that receives fees for aggregating the order.
oracleAddress	address	Address of oracle contract that conforms to the bZx Oracle interface.
loanTokenAmount	uint256	Total units of the loanToken that will be loaned.
interestAmount	uint256	Amount of interest that will be paid to the lender in interestToken units per 24 hour period the trade is opened.
initialMarginAmount	uint256	The minimum percentage margin amount required for a trader to receive funding for their loan and for them to open new trades after receiving funding.
maintenanceMarginAmount	uint256	The margin percentage amount at which a trader's position is force liquidated if their initial margin falls to this level. This also returns the loanToken to the lender and ends the loan.
lenderRelayFee	uint256	Total units of protocol token (BZRX) the lender pays to feeRecipientAddress.
traderRelayFee	uint256	Total units of protocol token (BZRX) the trader pays to feeRecipientAddress.
expirationUnixTimestampSec	uint256	Time at which the lending order expires (seconds since unix epoch).
salt	uint256	A pseudo-random 256-bit salt to ensure a unique loan order hash.
signature	bytes	ECDSA signature of the above arguments.

The Order Object

Initialization

• Lender calls approve function authorizing bZx to move the loan token and BZRX token.

• Trader calls approve function authorizing bZx to move the collateral, interest, and BZRX token.

• Both lenders and traders can serve as maker or taker of a loan order.

Execution

1. The maker creates an order object using the bZx Portal or via a relay.

2. The order object is signed with maker's private key to ensure the order cannot be altered.

3. The maker broadcasts the order object to relay or intended recipient via an arbitrary medium.

4. The taker intercepts the signed order object, presenting it to the bZx contract while passing parameters specifying the collateral token and quantity of the token to be borrowed, if not previously specified by the maker of the order.

5. The bZx contract verifies the maker's ECDSA signature, the order parameters, and the order expiration, before moving the loan token and required collateral token to escrow in *bZxVault*. The interest owed over the lifetime of the loan is calculated and also moved into escrow in *bZxVault*. If the loan is canceled early, the remaining interest and collateral tokens revert back to the trader. Multiple traders can fill a loan order, until all "loanTokenAmount" is taken. This means that the bZx protocol allows for partial fills.

Timeline

2017

JUNE

Kyle J. Kistner and Stani Kulechov begin brainstorming decentralized margin lending solutions while discussing ETHLend.

JULY

Tom Bean and Kyle J. Kistner begin discussing cryptocurrencies and margin lending, exploring the potential for a collaboration.

AUGUST

05 Tom Bean and Kyle J. Kistner conceive bZx, originally naming it broker0x. Whitepaper development begins.

SEPTEMBER

29 First Github commit in the protocol contracts code repository.

DECEMBER

20 Adrian Li joins the development team. Development of the bZx portal begins.

2018

JANUARY

10 Development of the bZx.js library begins.

12 The bZx website launches.

25 bZx sponsors ETHDenver, attracting initial attention.

FEBRUARY

03 bZx officially announces. "Welcome to b0x" published on Medium.

11 bZx officially releases the whitepaper after five months of concurrent coding development.

13 honestbonsai joins the development team, assisting in development of the bZx.js library.

18 bZx debuts at ETHDenver.

MARCH

15 bZx debuts as Title Sponsor of TokenFest in San Francisco.

APRIL

20 bZx protocol debuts its fully functional smart contracts onto the Ropsten Testnet

24 bZx portal is released. Order object creation and order filling are functional.

Manish Singh and Stani Kulechov join Advisory Board.

MAY

01 bZx presale begins

15 The bZx portal allows users to fill 0x orders and make trades using KyberNetwork. Lenders can now monitor open loans.

25 The bZx.js library is finished, with all sections of the portal complete. The bZx portal now allows bounty hunters to monitor and liquidate bankrupt margin accounts.

JUNE

Security audits with zk-labs begins.

Purple Paper Development (a reference implementation in Haskell) begins

bZx launches full website. Closed beta mainnet release of the bZx protocol.

JULY

Development of iToken contracts begin.

AUGUST

Development of the bZx wallet SDK.

SEPTEMBER

Token governance development

OCTOBER

Oracle marketplace integration.

NOVEMBER

Cross-chain asset lending development begins.

References

[1] Open Zeppelin. 2017. On the Parity Multi-sig hack.
https://blog.zeppelin.solutions/on-the-parity-wallet-multisig-hack-405a8c12e8f7

[2] Parity. 2017. The Multi-sig hack: a post-mortem.
https://blog.ethcore.io/the-multi-sig-hack-a-postmortem/

[3] Request Network. 2017.
https://github.com/RequestNetwork/RequestVesting

[4] Vitalik Buterin. 2017.
https://twitter.com/VitalikButerin/status/911218043761573889

[5] Jack Peterson. 2017.
Missing links in the Ethereum development stack
https://www.youtube.com/watch?v=FPHXbJPVVaA

[6] Will Warren & Amir Bandeali. 2017. 0x: An open protocol for decentralized exchange on the Ethereum blockchain.
https://0xproject.com/pdfs/0x_white_paper.pdf

[7] Loi Luu. 2017. KyberNetwork.
https://kyber.network/assets/KyberNetworkWhitepaper.pdf

bZx

Margin. Decentralized.

bZx.network t.me/b0xnet @b0xNet