Development Roadmap

Timeline

2016

- **Q4**
    - ✓ 0x founded
    - ✓ Deployed pre-alpha contracts to Ropsten testnet

2017

- **Q1**
    - ✓ Released whitepaper
- **Q2**
    - ✓ Hired Fabio
    - ✓ Deployed alpha contracts (upgradeable) to Kovan testnet
    - ✓ Launched 0x OTC on Kovan
    - ✓ 1st Place, Consensus 2017 Startup Competition
    - ✓ Hired Leo
    - ✓ Released 0x.js v0.8.0
- **Q3**
    - ✓ Contracts moved from alpha to beta via governance on Kovan
    - ✓ Hired Alex
    - ✓ Hired Ben
    - ✓ Token sale announcement
    - ✓ Security audits completed
    - ✓ 0x v1.0.0 deployed on mainnet
    - ✓ ZRX token sale
    - ✓ Hired Phillipe
    - ✓ Radar Relay beta launch
    - ✓ 0x.js updates
    - ✓ 0x Portal announcement
    - ✓ Standard relayer API draft published
- **Q4**
    - ○ 0x hackathon
    - ○ 0x external development grant program
    - ○ Relayer legal framework
    - ○ Website redesign
    - ○ 0x Portal feature updates
    - ○ Standard relayer API v1 finalized
    - ○ 0x.js v1.0.0 released
    - ○ Protocol features and optimizations R&D
    - ○ 0x v1.x.x deployed on Kovan and Ropsten testnets

2018

- **Q1**
    - ○ Governance R&D
    - ○ Reusable relayer UI components

    - 0x v2.0.0 deployed on mainnet
- **Q2**
    - Trade explorer v1 released
    - Governance whitepaper

Core functions

1. Developer tools
    a. 0x.js v1.0.0
    b. 0x.js Tutorials
    c. Contract ABI Documentation & Tutorials
    d. Standard Relayer API
2. User tools
    a. ERC20 Token Wallet
    b. Token Explorer (via TokenRegistry)
    c. Trade Explorer
    d. Governance Web Application
3. R&D
    a. Governance Protocol
    b. ZEIP Process & Contribution Guidelines
    c. Privacy Preserving Cryptography
    d. Decentralized Order Book

Overview

The 0x team will be responsible for the development of open source software, tools and infrastructure that support 0x protocol, its users and the surrounding ecosystem. 0x will be funded through the sale of ZRX tokens (0x protocol's native token) in a public token launch to be carried out on the Ethereum blockchain. Activities may be broken down into three primary areas: user tools, developer tools and R&D.

	User Tools	**Developer Tools**	**Research & Development**
Activities	- User Interfaces - Protocol Interaction - Decentralized Gov. - Order Inspection - Trade Explorer - Documentation	- 0x.js - HTTP/WS wrapper - Relayer Tools - DB + Websockets - Standard API - UI components - Trader tools - Relayer aggregator	- Governance - Blockchain Abstraction - Crypto Abstraction - Future Proofing - Privacy - Scaleability - Decentralized Orderbook - ZEIPs
Team Members	- Product Manager - Project Manager - UI/UX Designer - Frontend Engineer - Blockchain Engineer	- Product Manager - Product Marketing Manager - Project Manager - API Developer - Blockchain Engineer	- Research Director - 2x Technical Staff

User Tools

0x OTC

0x OTC is a browser-based web application that facilitates over-the-counter exchange using 0x protocol. 0x OTC allows users to generate, inspect and fill orders that adhere to 0x message format. Makers must manually share their orders with a known counterparty (email) or broadcast their orders across their social networks (Twitter, Reddit, Facebook). 0x OTC will not act as a Relayer that hosts and maintains a real-time order book, it will merely allow users to leverage the message format. 0x OTC is open source and free to use.

Decentralized Governance

Web application that allows stakeholders to generate, inspect and vote on proposed updates to the protocol. It may be possible to integrate with Aragon or Boardroom, though we will need to customize the UI and governance logic for 0x protocol.

Documentation & Tutorials

Documentation for non-technical users that explains how 0x protocol works. Demonstrate functionality with visual aids, tutorials and examples.

Trade Explorer

All information on the Ethereum blockchain is public. However, it is much easier for an experienced programmer to interact with and monitor the blockchain than a non-technical end user. In order to reduce this information asymmetry, we will create a trade explorer that compiles and presents data related to trading activity on 0x protocol. Examples of information the trade explorer will present include:

- trading statistics and history associated with each ERC20 token, Relayer and dApp
- trading statistics and history associated with individual addresses
- performance leaderboards
- integrate identity/reputation systems

Token Registry

Verify ERC20 token addresses and exchange rates using trusted information provided by the Token Registry. Allow users of 0x protocol to input an order JSON and get out a human-readable visual representation of the order and its associated parameters.

Metadata

Each token within the TokenRegistry is assigned a 34-byte ipfsHash, which is a self-describing content-addressed identifier that maps to a hash-chain based data structure stored on IPFS. Token metadata is then expressed within a JSON document, as per the IPLD standard (see the spec), with named merkle-links that can be traversed.

ERC20 Token Wallet

The ecosystem needs a user friendly wallet for tokens that makes it easy to interface with all of the standard token functionality. We plan on integrating this wallet with 0x OTC in order to make trading tokens with a known counterparty as simple as a regular token transfer. It will also allow users to interact with the 0x Token Registry to query and verify information about tokens in the ecosystem.

Developer Tools

0x.js

Official JavaScript Library for 0x Protocol. Provides an API for interacting with and listening to 0x smart contracts, ERC20 tokens, and converting between Ether and ERC20 wrapped Ether tokens. Version 1.0.0 will also listen for and handle pending transactions and blockchain reorgs.

Standardized Relayer APIs

If relayers adopt a standard format for their public APIs, it will reduce friction costs to trade across exchanges. This would also allow us to build tools that allow connectivity to any relayers that utilize this format. We will create and strongly recommend that relayers use this standard API.

Relayer Aggregator

With a standard relayer API, it will be easy to aggregate relayer orders into a single data stream. This will allow traders to programmatically interact with a single global liquidity pool.

0x Components

We will create reusable and customizable UI components for relayers, which will simplify the process of creating a prototype front-end.

Research & Development

Future-Proofing

The rapid pace of innovation in cryptography, smart contracts and decentralized networks practically guarantees that current standards will soon be outdated. As developers and maintainers of 0x protocol, it is our job to ensure that new innovations may be seamlessly integrated. Key components of 0x protocol that may need to be replaced over time include hashing algorithms (see multihash, Blake2b), cryptography (non-secp256k1 elliptic curves used in secure hardware enclaves, zkSNARKs, BLS, Schnorr signatures, ...) and smart contract integration capabilities. SIMD operations in the EVM.

Decentralized Governance

No reason to use a simple majority vote when there is an entire design space to work with. How do we maximize security while minimizing the chances of the protocol being held hostage by a small minority? Are commitments needed to submit proposals? Perhaps vetoes are more important than affirmative support?

Privacy

All trading activity through 0x protocol's shared settlement layer is public. While transparency can be healthy for markets, there are legitimate reasons why market participants would want privacy. Part of our research and development efforts will go towards extending 0x protocol to new and emerging cryptographic primitives such as zkSNARKs to allow for fully trustless and untraceable exchange. Potential approaches for driving this research effort could include:

- Provide funding to Alessandro Chiesa's (UC Berkeley) or Eli Ben-Sasson's (Technion) research group to complete a summer research project (short term) or bring on a new PhD student (long term).

Decentralized Order Book

For now, individuals will act as Relayers responsible for hosting and maintaining order books. Relayers will likely use conventional centralized databases. In the future, we would like to develop a Relay protocol that moves the order book from a centralized database onto a decentralized low-latency peer-to-peer network where swarms of nodes are compensated for propagating orders across the network according to the level of contribution made. Potentially based upon the IPFS tech stack, something like orbit-db.

0x Improvement Proposals (ZEIPs)

ZEIP1: Support Order Generation for Smart Contracts
ZEIP2: Order Matching
ZEIP3: Add Support for Cross-Relayer Orders
ZEIP4: Add orderType Field to Message Format
ZEIP5: Standard Deposit Contract
ZEIP6: Cross Chain OTC Trades with EVM Blockchains

EIP50: Struct Support in the Contract ABI

EIP50 proposes an extension to the contract ABI to support structs passed in as arguments to external functions. This would allow us to establish a standard *OrderSchema* data structure that conforms with 0x message format, simplifying the contract interface for Exchange.sol. A standard *OrderSchema* struct would not only improve code readability but also simplify external contract interactions.

EIP101: Currency & Crypto Abstraction

EIP101 includes a proposal to move Ether up one level of abstraction, allowing Ether and other Ethereum-based sub currencies (ERC20 tokens) to be treated similarly by smart contracts. For now, a “wrapper” smart contract may be used to extend the ERC20 token interface to deposited Ether. For reference, see the Maker implementation or Gnosis implementation.

ERC223 Token Standard

ERC223 allows a token transfer to contain an arbitrary data parameter that may be called by the receiving contract. This essentially allows tokens to have “push” functionality rather than “pull” functionality only. This would remove the need for the taker of an order to grant a token allowance to the Proxy contract, as they could transfer a token to the Exchange contract and while calling the fill function at the same time.