OAX Foundation –
OAX Parachain Technology Paper

March 2020

Overview

Today's OAX ERC20 token on its own provides limited utility. The development plans of the OAX Foundation aims to remedy that problem by building OAX's own blockchain, built on Substrate as part of the Polkadot network.

The OAX Parachain will offer features such as equivalent ERC20 functionality, atomic swaps, fee delegation and batched multiple transfers. These features can be used to build Decentralized Finance (DeFi) protocols to attract DeFi DApp development. The first protocol we will target is for Swap DEXs.

Goals

1. Build the OAX chain on the Polkadot network.
2. Provide chain differentiation by implementing atomic swaps, fee delegation, multiple transfers and a token standard like ERC20.

Requirements

OAX Blockchain Features

The OAX Foundation will achieve its development goals by building an OAX Parachain using Parity's Substrate technology. To differentiate the OAX Parachain with a vanilla Substrate chain, we plan on adding the following features:

1. ERC20 functionality;
2. Atomic Swap;
3. Fee Delegation;
4. MultiTransfer.

1. ERC. 20 Functionality

To support DeFi use cases such as lending or swaps, we need to be able to represent other tokens on the OAX Parachain. This is akin to ERC20 on Ethereum or eosio.token on EOS.

ERC20 Functions

There are several ERC20 implementations available in the community such as the ones from OpenZeppelin and Consensus. There is also a proof of concept ERC20 implementation for Substrate available here.

Please refer to them for reference.

Message signatures:

function totalSupply()
function balanceOf(address)
function mint(address, amount)
function burn(address, amount)
function transfer(receiverAddress, amount)
function transferFrom(senderAddress, receiverAddress, amount)
function approve(spenderAddress, amount)
function increaseApproval(spenderAddress, amount)
function decreaseApproval(spenderAddress, amount)

*increaseApproval and decreaseApproval are methods introduced to mitigate the situation where an approver changes their mind and decides to reduce the amount approved by sending a subsequent approve(). The approvee can front run this by seeing the reduced approval and spend tokens (setting high gas fee for this transaction), then spending the newly set approval again.

2. Atomic Swap

Traditional blockchains have an API to transfer tokens between wallets. Swaps involve two transfers and atomic swaps are done to ensure that both legs of a swap are guaranteed. Atomic swaps are done using smart contracts involving the swap of two distinct tokens. In the case of a swap between two ERC20 tokens, the chain for both tokens is Ethereum but it still requires two transactions.

A native swap SRM would provide the same guaranteed execution but in one transaction instead of two.

Note that this Atomic Swap implies that we will have implemented ERC20 functionality tokens on our parachain. This swap would not work for a swap between OAX Parachain and another Polkadot chain (until we find out more about how Polkadot interoperability via bridges work).

> **API signature:**
> *atomicSwap(tokenA, walletAddressA, sigA, amountA, tokenB, walletAddressB, sigB, amountB)*

This function requires signatures from both parties. Party A signs the atomicSwap message (setting an expiry) and passes message to Party B to sign and submit to the blockchain.

To prevent Party A's offer from living indefinitely, the user is able to set the message to expire at a certain time. The expiry field works to prevent Party B from holding the signed message for an unreasonable amount of time before trying to perform a swap on the offer.

As a bonus to better showcase this Atomic Swap feature, we may also create a demo UI that allows users to create/accept swap offers offline by simply exchanging a message (or QR code).

Atomic Swap Tests

1. Swap fails if either party has insufficient balance, leaving original token balances unchanged.
2. Swap succeeds and both token balances are updated within the same blockchain block.

3. Fee Delegation

Fee delegation allows the gas cost of a transaction to be paid by someone other than the user.

This feature eases adoption for DApps because users do not need to have native OAX Parachain tokens in order to transact with the OAX Parachain. Users not having to have a particular token to pay for gas removes a significant hurdle for DApp adoption. Imagine going to a DEX to trade your “ABC” token for “XYZ” token; and then finding out you also need an unrelated third party token to pay for the trade? The fee delegation function can be used to remove this requirement, which simplifies the process for users to use the DApp.

DEX operators could also use this function to provide no-fee limit orders but charge fees for market orders; encouraging participants to add liquidity rather than taking it away.

Blockchain transactions need to support a second signature field. If a third party (typically the DApp operator) wishes to pay the fee for a transaction, they must sign the newly introduced “gas payer signature” field. To do so, we must extend Substrate’s Extrinsic data type (Rust trait) to support a second signature.

Fee Delegation Tests

1. Third party pays the fee for a users' transaction by signing the gas payer field. The transaction is successful and the gas fee has been paid by a third party instead of the user.
2. Third party signs a delegated fee transaction but does not have the funds to pay. The transaction should fail as the user should not be expected to pay an additional fee.

4. MultiTransfer

The introduction of a multi transfer function will allow developers to batch multiple transfers into one for cheaper gas costs. The idea is that the message size of this batched transaction should be smaller due to fewer repeated fields (such as the from address which would be identical for each inner transaction), resulting in lower gas cost.

It should be noted that this is NOT an atomic transaction as we do not ensure that all transactions are successful or would otherwise then be rolled back. Thus, under this MultiTransfer functionality, some inner transactions could still fail but the others would remain committed.

MultiTransfer Tests

1. Create transaction with multiple transfer transactions. If any transactions are mined, the transaction status should be SUCCESS.
2. Create a multi transfer with insufficient funds. The returned status should be FAIL.

Conclusion

While we're still in early stages with Polkadot, and the building of the OAX Parachain, the team believes the proposition aligns with the work that has been done since the founding of OAX. Between the vision that Polkadot has, to the strong community that has driven its progress so far, we look forward to bringing additional value and playing our role in the addressing the shortcomings of the digital asset industry: speed, scalability, trust, and now, interoperability.