Set: A Protocol for Baskets of Tokenized Assets

Felix Feng, Brian Weickmann

v1.0 - Revised

September 13, 2018

Abstract

We present a protocol ("Set") that facilitates the creation, issuance, redemption, and rebalancing of fungible, collateralized baskets of tokenized assets ("Set tokens" or "Sets"). The protocol employs an off-chain order relay and on-chain settlement in which users can issue new Sets that utilizes liquidity from decentralized exchanges. Users who desire to track an index or have their portfolios automatically updated can subscribe to a Rebalancing Set Token. The protocol is open, non-rent seeking, and aims to utilize decentralized governance.

Contents

1 Introduction

1.1 The Tokenization of the World

With the creation of the Bitcoin blockchain and subsequently the Ethereum blockchain, anyone can create their own digitized token to represent securities, goods, services, real world assets, etc. As of August 2018, there are 1,900 tokens listed on CoinMarketCap that represent over $230 billion in value, not to mention a long-tail of obscure tokens not tracked by CoinMarketCap [1]. Tokens are being created for numerous purposes, including but not limited to:

- **Currency on Blockchains / DAGs**: Blockchains (e.g. Bitcoin, Ethereum, NEO, Zcash), directed acyclic graphs (e.g. IOTA, HashGraph), and their forks (e.g. BitcoinCash, Ethereum Classic) have their own native tokens that incentivize parties to process transactions.
- **Second Layer Protocols**: Protocols such as 0x Project [2], Kyber Network, and Raiden are issuing utility tokens that are used to transact on their networks and support decentralized governance.
- **Digital Goods**: Gaming communities, streaming services, content platforms, and gambling sites are using tokens to represent virtual goods, incentivize direct payment of content creators, and create economic systems that bypass traditional intermediaries.
- **Decentralized Financial Products**: Projects such as dY/dX [3] and Dharma [4] are allowing anybody to issue options, short sells, and loans that are represented by tokens. MakerDAO, Basecoin, and Fragments are creating "stablecoins" that aim to resist fluctuations in value. Projects such as Augur and Gnosis allow anyone to interact with predictions markets.
- **Real World Assets**: Projects such as TrustToken and DigixDAO are bringing the tokenization of real-world assets.
- **Securities**: Blockchain Capital, a cryptocurrency hedge fund, has issued tokenized securities to represent stakes in their fund. Harbor and Polymath are building platforms for parties to issue tokenized securities and investors to compliantly trade securities on secondary markets. Numerai has taken a different approach, awarding data scientists with profit-share tokens for their work in generating profits for the hedge fund.

We think that, as more people and organizations turn to tokens for economic coordination in the coming decades, millions or billions of different tokens could exist.

1.2 The Pains of Increasing Token Cardinality

There is growing complexity and pains in dealing with an ever-growing number of tokens. We attribute this to the lack of a standardized protocol for multi-token use cases. Some of these pains are detailed below:

- **Transaction Costs**: Existing transaction infrastructure was not designed with multi-token use cases in mind. External accounts (standard addresses in Ethereum) need to construct, sign, broadcast, and pay a transaction fee for each token transfer. The transaction fees and effort of dealing with a multitude of tokens grow linearly with the number of tokens transacted. If one wants to send 100 different tokens using a standard Ethereum external account, one would need to sign and broadcast 100 transactions and thus pay 100 transaction fees. This limits the scalability of decentralized apps and results in a subpar user experience.

- **Cognitive Overload**: As the number of tokens that users need to interact with grows, the cognitive load required for management of tokens follows suit. For each token they own, users must understand each token's qualities (e.g. price, monetary policy, utility, liquidity, interface, etc). Each token needs to be treated individually, and users must retain everything in their head at once.
- **Client Limitations**: Ethereum clients such as Parity, Mist, and Metamask are limited in their ability to manage groups of tokens. Clients don't have features that allow for the grouping together of tokens or batch transfers. Thus, users have no option but to manually and repeatedly perform common operations like viewing, transferring, and function calls across multiple token smart contracts.
- **Investor Pains**: Investors who hold a multitude of tokens lack the tools to actively acquire, manage, and reassess their investments. Standard active portfolio management practices such as rebalancing, bucketing, and transfers become onerous and repetitive processes. Analyzing groups of tokens requires individually aggregating the prices of tokens, and bucketing tokens must be done manually. These inconveniences even have financial repercussions when one considers how fees accumulate as a result of token-by-token exchange trades and withdrawals.
- **Developer Pains**: Developers lack a standard for hiding the details of multiple tokens from users on the protocol level. Currently, information hiding must be done on the application-layer of the stack vs. natively in the protocol layers. Developers also need to understand all the tokens that their system deals with. Finally, it is difficult to create a good user experience to onboard new users if the users are inundated by the details of tokens.

1.3 Abstraction as a Solution

In software engineering and computer science, abstraction allows programmers to think on a certain level of complexity while hiding away details not relevant to the problem at hand. We use abstractions to prevent overloading the end user with details when they care more about higher level concepts. This lets developers focus user attention on what objects represent within their platforms, rather than their nuts and bolts.

In traditional finance, we have abstractions like the American stock market indices (e.g. S&P 500, Dow Jones Industrial Average (DJIA)) that represent hundreds or thousands of individual stocks. In the insurance industry, we purchase policies comprised of a set of services in exchange for paying an insurance premium, without the need for fretting over individual coverage scenarios.

For cryptocurrencies, we can envision an abstract token or meta token, a single token representing a basket or portfolio of its underlying tokens.

2 Set Token

2.1 Overview

We introduce a Set Token ("Set"), an abstract, fungible token fully collateralized by its underlying tokens. While Sets will eventually interoperate across blockchains, Sets are described in this document as smart contracts on the Ethereum blockchain that adhere to the ERC20 token standard.

2.2 Properties

- **ERC20 token**: Sets comply with the ERC20 standard[5], an interface of functions and events that an Ethereum token contract must implement. The ERC20 standard includes functions such as transfer, approve, totalSupply, and balanceOf. This means that Sets can be transferred, traded, and approved to transfer: just like any other ERC20 token.
- **Collateralized**: Sets are collateralized by their underlying tokens (components). This means that the Set trustlessly has custody of the components and can only be accessed through its exposed

methods. Sets are created by transferring the underlying tokens to Set's Vault and issuing a new Set Token representing those tokens.

- **Redeemable**: Sets can be redeemed or traded for their components. As Sets are backed by the underlying tokens, users can be certain that their Set can be traded for their components.
- **Specified tokens and units**: Sets are specified by a list of underlying tokens and their respective quantities. Each token minted from a Set contract cannot deviate from the specified tokens and ratios as defined during the construction of the Set contract. As long as the desired tokens issued or redeemed matches the predefined Set weights, it is possible to issue and trade fractions of Sets.
- **Composable**: As long as the Sets conform to the ERC20 standard, Sets can be composed of other Sets. This makes it possible to have a single token to represent a limitless number of other tokens without hitting the block gas limit.
- **Trustless**: Set is open source and functions only as programmed. Set has been designed so that no owners or administrators that can cause changes to the held collateral tokens.

2.3 Benefits

- **Save gas**: Without Sets, acquiring and transferring multiple ERC20 tokens requires paying transaction fees on transfers of each token. By transacting using Sets, users only need to pay transaction fees on a single transaction for the tokens they represent.
- **Focus on higher-level concepts**: Sets allow users to think about higher-level concepts. Since Sets are composable, it is possible to build up multiple layers of abstraction. This makes Sets a very generalizable and a powerful tool for building and reasoning about complex decentralized applications.
- **Underlying value**: The intrinsic value of a Set can be determined by summing the value of its underlying components. We expect that Sets will be priced at a premium on token exchanges relative to the cost of their underlying tokens, reflecting the cost of minting the Sets.
- **No Counterparty Risk**: Traditional "higher-level assets" such as ETFs often trade at a discount, because these assets are not without counterparty risk. Because a trustless, autonomous smart contract holds custody to the underlying tokens, there is no third party that can fail to live up to its contractual agreements.

3 Use Cases

3.1 Index Token/ETF

In the traditional financial industry, market indexes are collections of assets that track the overall movement of a specific theme or sector in the equity markets. They represent the aggregate value of their underlying securities. Investors usually use indices to track the value of the market over time, gauge the market's financial health, and benchmark their own returns. Some of the most widely cited market indexes include the S&P 500, which tracks the 500 largest companies having common stock on the largest US stock exchanges, and the DJIA, which tracks 30 large publicly owned companies in the US. For investors, index fund investments are a passive form of fund management and are considered ideal core portfolio holdings for many people's accounts. They usually have lower management expense ratios, allow for diversification, and lower trading costs for investors. In the cryptocurrency world, a Set is the natural tool for the representation of a collection of publicly traded tokens. As a market index that tracks tokens, Sets can be used for investment, tracking, market-making, and advanced derivatives:

- **Investment**: As investment vehicles, Sets offer a number of advantages, such as cost reductions, asset diversity, and lower maintenance requirements.

Without Sets, acquiring a basket of tokens can be prohibitively expensive in terms of fees and energy. Currently, investors who want exposure to a market index need to purchase the individual underlying tokens themselves. The investor would need to 1) register and provide personal identification information for numerous exchanges, 2) pay maker-taker trading fees for each token purchase, and 3) pay network transaction fees to withdraw/transfer each token into their personal wallet. With Sets, the investor could purchase a single representative token from a single exchange, greatly reducing trading fees and saving time.

Owning Sets allows investors to diversify and manage their risk across any particular asset class. Sets makes it easy for retail and institutional investors who do not know which individual crypto assets to purchase to get exposure to the entire market or one of its sub sectors. For example, an investor wanting exposure to the top 10 tokens by market capitalization for long-term investment can invest by purchasing a single Top10Set composed of the top 10 ERC20 tokens. For a desired Set that does not exist, the investor can invent/create their own using the protocol.

- **Tracking**: Sets are well suited to be a measurement of value for the token market. Sets can be used by investors and analysts to describe the market and compare returns on investments. Our current tools for tracking are quite limited. CoinMarketCap only provides an aggregate market capitalization of cryptocurrencies. Current tools are also unstandardized. Sets can represent a construct that the industry relies on for evaluating the health of cryptocurrencies and performing macro analyses.

- **Rebalancing**: Because Sets are passively managed and not meant to be actively traded, Sets are low-maintenance, only requiring infrequent rebalances. As the price of the underlying assets fluctuate, traditional market indices are rebalanced or updated by adjusting the collection of assets that are included in the index. This is a practice that is usually done quarterly, semi-annually, or annually. Set investors can easily rebalance their portfolios by trading for an updated Set that reflect the current collection of tracked tokens or issuing a Rebalancing Set. Rebalancing is discussed in detail in a subsequent section.

- **Market-Making**: Under the hood, Sets are most similar to exchange-traded-funds (ETFs). An ETF is a type of fund that owns the underlying assets and divides ownership into shares. Supply of ETF shares is regulated through creation (trading of underlying assets for ETF shares) and redemption (trading of ETF shares for the underlying assets). Creation and redemption involve very specialized investors known as authorized participants (AP). The AP assembles the required portfolio of assets and turns that basket into the fund for newly created ETF shares. Set issuance and redeem mechanics are synonymous with the ETF's creation and redemption, and thus there will be the equivalent of APs in this system that will be issuing and redeeming Sets.

  In the token world, traders can serve as liquidity-providers who are incentivized to normalize Set price deviation, preventing large deltas between the Set price and the Set's aggregate component value. Set's permissionless nature allow traders to take advantage of these risk-free arbitrage opportunities by issuing and redeeming Sets or by trading for other Sets.

- **Advanced Derivatives**: It is possible to create sophisticated financial products such as basket default swaps, collateralized debt obligations, or basket of futures by combining Sets with primitives such as derivatives and loans (e.g. dY/dX [3] and Dharma [4], respectively). These instruments allow investors to manage risk and better facilitate token price discovery.

3.2 Multi-Token Decentralized Applications

As more second-layer protocols go live, decentralized applications ("dApps") will be allowing complex operations that incorporate multiple protocols. Usage of these protocols usually requires protocol tokens, and users of these dApps would need to purchase tokens for each protocol operation. Thus, users may need to hold dozens of tokens for each of the various protocol operations. Sets are well suited to represent units of composite-work in multi-protocol applications.

For example, let's say a developer wants programmatically to process a file in a MapReduce fashion on computers all over the world using Golem and then store it in a decentralized manner using Storj. Using a decentralized exchange protocol such as 0x protocol, the developer can programmatically purchase a single {ComputeStore Token} which is composed of 100x Golem and 100x Storj tokens using wrapped Ether, ether that has been converted into the ERC20 specification. Then the developer can call the redeem function in the {ComputeStore Token} contract to get the underlying Golem and Storj tokens for usage on the Golem and Storj networks. Any unused tokens can be traded, sold, or issued into a new Set.

3.3 Decentralized Finance

There are numerous emergent, crypto-native use cases for Sets that have yet to be fully explored. Some of these use cases include:

- **Composite StableCoin**: A Set composed of other stable coins to smooth out volatility and diversify risk of de-pegging with even distribution across various approaches.
- **Protocol Collateral**: Complex Sets that are used for collateralization in second layer protocols (e.g. basket of goods for use as collateral in the MakerDao CDP System).
- **Hedged Bet**: A Set containing both a long position and a short position.
- **Fractional Ownership of Non-Fungible Assets**: A Set representing fractional ownership of a collection of CryptoKitties or other pieces of digital art.
- **Stake-able Portfolio**: Capture your life's assets or portfolio in a single Set that can be used as collateral in staking systems.

4 Architecture

4.1 Overview

Set defines procedures for creating, issuing, rebalancing, and redeeming Set Tokens using a collection of smart contracts and integrations with liquidity pools. Set's architecture has been inspired by 0x Protocol and dYdX. Set was designed to be a collection of modular smart contracts where additional components can be seamlessly integrated and easily upgraded.

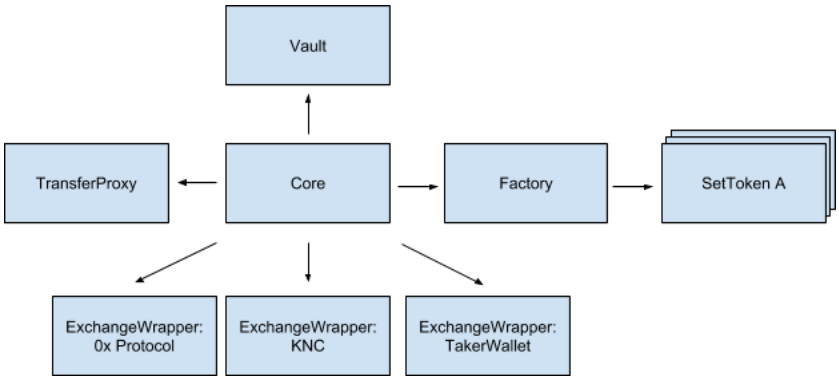

Figure 1: Contract Architecture Diagram

4.2 Smart Contracts

Set consists of a system of smart contracts [6] deployed on the Ethereum network. These include:

- **Core**: This public-facing smart contract contains all the business logic and external functions associated with creating, issuing, rebalancing, and redeeming Sets. Core serves as a kernel that orchestrates interactions between the various other smart contracts.

- **TransferProxy**: This smart contract facilitates the transfer of ERC20 tokens for all transactions. Users who wish to utilize their tokens must authorize the TransferProxy contract to move their tokens by calling the ERC20-compliant token's approve function. The design of a centralized TransferProxy contract is intended to minimize the number of overall number of *ERC20 approve* transactions required in the system.

- **Vault**: This smart contract stores all the Set's component and contains mappings to track ownership of assets. Vault's accounting interfaces are only available to Core.

- **Factory(s)**: This smart contract contains the template for a specific type of Set and facilitates the creation of all Set token contracts. Creators can create a new Set token by calling Core's create function and supplying the Factory address and the required parameters. For a Factory to be valid, it needs to be registered with Core. Initially, there will be a SetTokenFactory (creating standard Sets) and RebalancingTokenFactory (a factory for creating Rebalancing Set tokens).

- **Set Token(s)**: This ERC-20 compliant smart contract represents a unique Set with specified ERC-20 components and units and tracks balances of the Set tokens. Each Set token smart contract is created by a Factory and is tracked by Core. Each smart contract exposes mint and burn functions that only Core could call during issuance and redemption.

- **ExchangeWrapper(s)**: This smart contract serves as a conduit between decentralized exchanges and Core to facilitate component-acquisition in an issuance order. Each exchange wrapper contract exposes a common exchange function that accepts a bytes string that encodes exchange messages. The ExchangeWrapper is responsible for parsing call data conforming to an exchange's interface ("exchange messages") and executing each order as the taker of each transaction. Initially, we will be supporting withdrawing tokens from the submitter's wallet, 0x Exchange V2, and Kyber Network.

4.3 Participants

There are a number of participants in the network: Creators, Users, Relayers, and Traders:

- **Creators**: Utilize their domain knowledge, creativity, and intuitions to design, create, and deploy Sets that have market appeal.

- **Users**: Can either 1) gather components from various centralized and decentralized liquidity sources or 2) create signed issuance order messages to issue Sets for trading, holding, and usage. They can also redeem their Sets for the components.

- **Relayers**: Host order books that facilitate issuance order matching, generating compensatory fees for their services.

- **Traders**: are economically-driven participants who compete to profit by gathering exchange messages and filling issuance orders as well as facilitating price discovery by issuing and redeeming undervalued/overvalued Sets.

5 Variant ERC20 Standard Considerations

5.1 Token Pausability

Many Ethereum ERC20 tokens implement "Pausable" functionality, which allows the creator of the token to prevent approvals, transfers or transferFrom functions from executing properly. Some tokens that implement Pausable functionality include EOS, Tron, Icon, Status, and Augur. Since Set's redeem operations requires that every single transfer to complete properly, a single token transfer failure could hold a Set's remaining

components hostage.

Set circumvents this risk by separating redemption (burning the Set and attributing components to the user in the vault) and withdrawal (transferring the tokens from the Vault to the user) into two separate steps. With this design, users will invariably be able to retrieve a Set's remaining unpaused tokens from the Vault.

5.2 Variant ERC20 Decimals

Although ERC20 tokens have standardized around 18 decimal places, many teams have decided to implement their own choice of decimal places. Some of these include Airswap (4 decimals), Zilliqa (12 decimals), and RChain (8 decimals). Decimal place variance makes it difficult to create Set Tokens that target a specific price point because tokens with lower decimal values have significantly more valuable base units.

5.2.1 Natural Unit

To solve this issue, Set introduces the concept of natural unit, which is the atomic, least divisible unit of a Set that can be issued or redeemed. Set creation enforces that the natural unit is larger than the transferable unit of the token with the smallest decimals place. This is represented by:

$$MinimumNaturalUnit = 10^{18-min(componentDecimals)} \tag{1}$$

The ideal natural unit that you use in Set creation is based on a number of parameters including:

- **Desired Price**: The price of a single unit (1×10^{18} base units) of a Set
- **Component Token Decimals**: The divisibility of the ERC20 component denoted by the decimals field
- **Component Token Price**: The price of a single unit of a component
- **Set Composition**: The desired portfolio allocation of the components in the Set

Example: Assume that a user wants to create a Decentralized Exchange Token (DEX) that contains Airswap (AST) and 0x (ZRX) tokens in a 50/50 allocation priced at $10. Furthermore, assume that AST is trading at $0.10 and ZRX is trading at $1. In order to achieve a 50/50 split of AST and ZRX, each Set needs 50 AST tokens (5×10^5 base units of AST) and 5 ZRX tokens (5×10^{18} base units of ZRX).

The problem arises when a user issues partial amounts of DEX Set Tokens, which have 18 decimals. The formula for calculating the amount of component token, i, required for an issuance is as follows:

$$amountRequired_i = \left(\frac{quantity_{issue}}{quantity_{Set}}\right) \times componentUnits_i \tag{2}$$

Where $quantity_{Set}$ is the base unit amount of 1 Set Token (10^{18}). If a user wants to issue 10^{12} base units of the DEX Set Token, the ZRX amounts work as expected:

$$amountRequired_{zrx} = \left(\frac{10^{12}}{10^{18}}\right) \times (5 \times 10^{18}) = 5 \times 10^{12} \text{ base units of ZRX} \tag{3}$$

However, the same math with AST leads to an untenable number:

$$amountRequired_{ast} = \left(\frac{10^{12}}{10^{18}}\right) \times (5 \times 10^{5}) = 0.5 \text{ base units of AST} \tag{4}$$

Since base units are not divisible and all decimals are rounded down, in order to issue this Set the user would "transfer" 0 base units of AST to collateralize the Set Token. However, the system would still credit the user with the collateral for the Set Token, in essence leaving the system not fully collateralized. Natural

units help fix this attack vector by requiring all issue amounts to be greater than the natural unit. Adding the natural unit changes Equation 2 to the following:

$$amountRequired_i = \begin{cases} \text{revert}() & quantity_{issue} < naturalUnit_{Set} \\ \left(\frac{quantity_{issue}}{naturalUnit_{Set}}\right) \times componentUnits_i & quantity_{issue} \geq naturalUnit_{Set} \end{cases} \tag{5}$$

The natural unit can be calculated using Equation 1, which returns 10^{14}. Additionally, the components units must be updated to target the correct amount of base units per Set Token (5×10^{18} ZRX and 5×10^5 AST):

$$5 \times 10^{18} = \left(\frac{10^{18}}{10^{14}}\right) \times componentUnits_{zrx} \tag{6}$$

$$componentUnits_{zrx} = 5 \times 10^{14} \tag{7}$$

$$5 \times 10^5 = \left(\frac{10^{18}}{10^{14}}\right) \times componentUnits_{ast} \tag{8}$$

$$componentUnits_{ast} = 50 \tag{9}$$

Thus, for every 10^{14} base units of the DEX Set Token issued, the user must contribute 5×10^{14} ZRX base units and 50 AST base units. With this configuration, the target specifications are achieved, one DEX Set Token equals \$10 and has a 50/50 allocation split.

However, this equation is not complete, imagine the scenario where a user attempts to issue 1.05×10^{14} base units of the DEX Set. Again, the ZRX amount is valid, however the AST amount leaves the ability for a user to not fully collateralize their issuance:

$$amountRequired_{zrx} = \left(\frac{1.05 \times 10^{14}}{10^{14}}\right) \times (5 \times 10^{14}) = 5.25 \times 10^{14} \text{ base units of ZRX} \tag{10}$$

$$amountRequired_{ast} = \left(\frac{1.05 \times 10^{14}}{10^{14}}\right) \times 50 = 52.5 \text{ base units of AST} \tag{11}$$

A further restriction must be put on the issuance amount, in this case the issuance amount must also be a multiple of the natural unit. This guarantees that the component unit amount in Equation 5 will always be multiplied by a whole number, thus guaranteeing the required amounts are always whole numbers. The final equation for calculating required units for component token i, using the natural unit, is as follows:

$$amountRequired_i = \begin{cases} \text{revert}() & quantity_{issue} \mod naturalUnit_{Set} \neq 0 \\ \left(\frac{quantity_{issue}}{naturalUnit_{Set}}\right) \times componentUnits_i & quantity_{issue} \mod naturalUnit_{Set} = 0 \end{cases} \tag{12}$$

5.3 Variant Transfer Receive Quantities

Although many ERC20 tokens decide to implement standard transfer and transferFrom functions, certain ERC20 tokens contain variant implementations of transfer and transferFrom functions where there is an asymmetry between the amounts of tokens sent from an address and amount received. This is common in tokens that include fees in their transfer and transferFrom functions such as Havven's Stablecoin Nomins. This could result in transfers where fewer tokens are received by the Vault than expected.

To ensure full collateralization, Set enforces checks that the amount received and transferred is the amount the transfer specifies. This comes with a conscious trade-off that tokens with non-zero transfer fees will not be supported by the protocol.

5.4 Variant Transfer Return Values

The ERC20 specification outlines certain success return values for transfer and approval functions. However, some projects have decided to return different values or no value at all, which causes checks to make sure a valid transfer occurred to fail. In order to handle this, Set implements an ERC20 Wrapper that allows for null or true return values to evaluate to true for this specific subset of functions, instead of reverting on a null response. This still means that tokens returning a non-boolean/null value will revert and not be usable in the Set system.

6 Specification

6.1 Overview

Core exposes a number of operations you can perform on a Set. Initially, a Set is designed and created through the Core contract and its specified Factory. Sets are then funded or issued through the standard Issuance Flow (if the user has all the underlying components) or through an Issuance Order (if the user needs to pull in liquidity). Users who wish to have their Sets updated based on predefined settings can choose to do so by issuing a Rebalancing Set Token. Finally, a Set can be redeemed: burning the owner's Set token and returning the underlying components.

6.2 Create

Creation is the process of instantiating a new ERC20-compliant Set Token contract. These contracts are instantiated by Factories, which serve as the blueprint for a Set Token contract to be constructed. Anybody can create a Set by calling the Core's create function by passing in the required parameters below:

Property	Type	Description
factoryAddress	address	The address of the factory to create from
components	address[]	The addresses of the component tokens
units	uint256[]	The amounts of each component token (per naturalUnit)
naturalUnit	uint256	The minimum unit to be issued or redeemed
name	bytes32	The bytes32 encoded name of the Set
symbol	bytes32	The bytes32 encoded symbol of the Set
callData	bytes	If required, additional data in bytestring format

There are no restrictions to how many different ERC20 tokens can be included, aside from block gas limits, and data input limits. Since Set Tokens are ERC20 tokens, Sets can be composed of other Sets. Deploying a Set Token creates an ERC20 token contract with 0 initial supply.
Initially, there will be two Factories that create the following:

- **Standard Sets**: Vanilla Sets whose composition and units have been pre-defined.
- **Rebalancing Sets**: Sets whose composition is programatically updated based on predefined rebalancing mechanics and inputs from a data source. We discuss Rebalancing in further detail in a later section.

In the future, new Factories can be introduced that create Sets with additional functionality and properties such as:

- **Regulatory Compliant Sets**: Sets with securities regulations, KYC/AML policies, and tax law compliance baked in, adhering to open standards such as R-Token or PolyMath.
- **Sets with Fees**: Sets where certain function calls impose small fees that accrue to the Creators or other specified parties, rewarding them for devising desirable compositions.

- **Sets of Non-fungibles****: Sets where tokens represent fractional ownership of collections of non-fungible tokenized assets that adhere to the ERC721 standard (e.g. art, title deeds, and sports teams).
- **Issue-Only**: Sets where the redemption feature has been disabled. Use cases may be securities of two companies that have participated in a corporate merger or acquisition.
- **Stake-able Sets****: Sets where the components can be contributed to staking pool protocols (e.g. Vest) or governance (e.g. Token Curated Registries).
- **Interest-Generating Sets****: Sets where components can be loaned out via flash lending protocols (e.g. Marble Protocol) or longer-term loans (e.g. Compound) to generate interest on idle components.

** Major protocol upgrades may be required to support this.

The creation flow is formalized as follows:

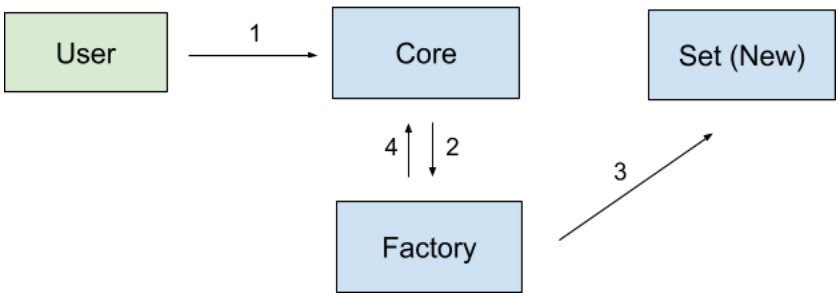

Figure 2: Creation Flow

1. User calls the create function on Core, providing as arguments the Factory address, component addresses, component units, natural unit, name, symbol, and other data (optional).
2. Core checks that the Factory address is authorized. If so, Core passes the argument data to the Factory.
3. The Factory deploys a new Set token smart contract with the provided parameters.
4. The Factory returns to Core the new Set address, which Core records in its mapping of tracked Sets.

6.3 Issue

Issuance is the process of minting new tokens for a particular Set, increasing the Set supply. With a valid deployed Set, anybody can call the Core's issue function to convert a specified mix of ERC20 tokens into a Set Token. For a standard issuance, users need to contain all components of a Set in a single public address and/or the Vault.

Issuance is a two-step process of depositing funds into the Vault and then issuing (attributing components to the Set and minting a Set Token to the user). Ahead of any deposits, users need to approve their tokens to the TransferProxy. Users can incrementally deposit tokens into their vault before issuing, as issuance uses a combination of the user's tokens in their wallet as well as their tokens in the Vault.

The issuance flow is formalized as follows:

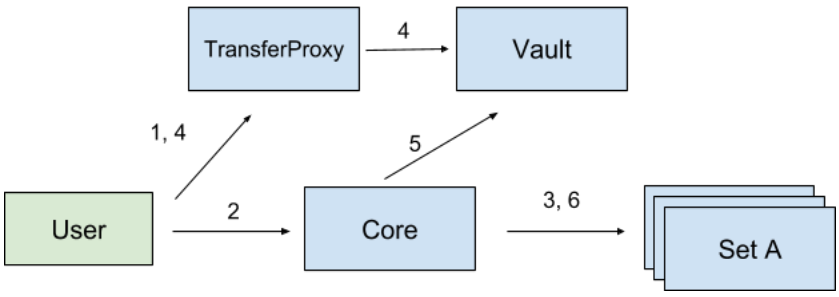

Figure 3: Issuance Flow

1. User approves components for transfer to the TransferProxy.
2. User calls the issue function on Core, targeting Set A and specifying an issue quantity. Core validates that the Set A is a known/tracked Set. The issue amount must be a multiple of the Set's natural unit.
3. Core reads the targeted Set's components, units, and natural unit.
4. For each component, Core calls the TransferProxy contract's transfer function and moves the funds from the user into the Vault.
5. The Core then increments the balance in the Vault for each of the transferred components to the Set.
6. When all the transfers are complete, Core calls Set A's mint function (callable only by the Core), which mints the corresponding number of Sets the user has issued.

6.4 Issuance Order

Acquisition of components for Set issuance is often a tedious process. Users must take multiple steps to aggregate components into a single address including: gaining access via various centralized (requiring KYC/AML) or decentralized exchanges, transferring components into a single address in the correct quantities, approving tokens to the TransferProxy, and calling issue function in Core.

Issuance orders is a sequence of steps that uses off-chain order relay and on-chain settlement pattern in which users can issue a Set using liquidity from various decentralized liquidity pools in a single step.

Users (Issuer) only need to generate and distribute signed issuance order messages off-chain that specifies their desired Set and the maker tokens they are willing to pay in exchange for the missing components of a Set. We suggest that users standardize around WETH or a stablecoin such as DAI and TrueUSD to improve liquidity.

Issuance orders are generally aggregated by Set Relayers who host the messages in a centralized order book for an agreed upon fee for Sets that have yet to be issued. These off-chain messages can be discovered by anybody and submitted to Core for fulfillment. To help facilitate issuances, it is in the best interest of Set Relayers to also offer services related to the aggregation of liquidity from decentralized exchanges and liquidity pools.

Traders can take advantage of arbitrage opportunities where there is a spread between the issuance order maker token value and the aggregate cost of exchange messages they are able to wrangle from various exchanges or relayers. Traders complete transactions by submitting an on-chain transaction to Core with issuance orders found on Set Relayers and exchange messages from decentralized exchanges. Unused maker token will be sent to the taker, representing the spread / arbitrage opportunity. To maximize profits, it is in the taker's best interest to aggregate orders that minimize the aggregate amount of taker token used.

Initially, Set will support ERC20 fills for 0x V2, Kyber Network, and from the Trader's wallet. Eventually, we are exploring supporting other liquidity sources and aggregators such as Uniswap, Totle, Bancor.

The order is filled in a single atomic transaction when a signed issuance order and a list of exchange orders are submitted to the Core contract. If the issuance order message is valid, the following sequence occurs in a single transaction:

1. The exchange orders are routed to their appropriate ExchangeWrappers where each order is sequentially executed utilizing the funds offered by the maker.

2. Utilizing the maker's existing components and the newly acquired tokens, Core mints a new Set token to the maker.

6.4.1 Message Format for Issuance Order

Issuance orders are data packets that are composed of the order parameters and an associated ECDSA signature. To produce the ECDSA signature, order parameters are concatenated, hashed to 32 bytes, and then cryptographically signed by the maker using their private key.

The order parameters are formalized below:

Property	Type	Description
setAddress	address	Set to issue
makerAddress	address	Maker of the issuance order
makerToken	address	Token the maker is paying in
relayerAddress	address	Relayer's address
relayerToken	address	The token paid to the relayer
quantity	uint256	Quantity set to issue; must be multiple of naturalUnit of set
makerTokenAmount	uint256	Amount in makerToken for the aggregate order size
expiration	uint256	Unix timestamp of expiration
makerRelayerFee	uint256	The maker relayer fee
takerRelayerFee	uint256	The taker relayer fee
salt	uint256	Random salt
requiredComponents	address[]	Components to be sourced
requiredComponentAmounts	uint256[]	Amounts of required components maker is looking to be sourced
orderHash	bytes	Keccak256 hash of above order data
v	uint8	ECDSA signature of the above orderHash
r	bytes32	ECDSA signature of the above orderHash
s	bytes32	ECDSA signature of the above orderHash

These data packets are generated off-chain and can be broadcasted to varying counterparties through any transport medium. Typically, these signed messages can be aggregated to Relayers, parties that host public order books where the messages will compete on terms.

6.4.2 Exchange Messages

Exchange messages are data packets that are included in the execution of a fill order. The requirement for valid exchange orders is that it fulfills the component tokens and their required quantities specified by the maker to successfully issue the Set.

Exchange messages are passed as order data that is parsed by the Core contract and its associated Exchange Wrappers. Only Exchange Wrappers that have been approved are valid.

Because the data requirements are different from each exchange, we encode the data in a bytestring with a common exchange-header. For example, 0x exchange orders accept a signed order, fill amount, and signature. While our Taker Wallet simply requires a token address and the token quantity. Developers can use our javascript library to format and encode the data into a format that is parsable by the associated exchange wrapper.

Exchange messages are grouped by exchange type, concatenated, and encoded in bytestring format. The design of the data packet has been inspired by networking protocol design. See below for a high-level diagram of the an example encoded bytestring:

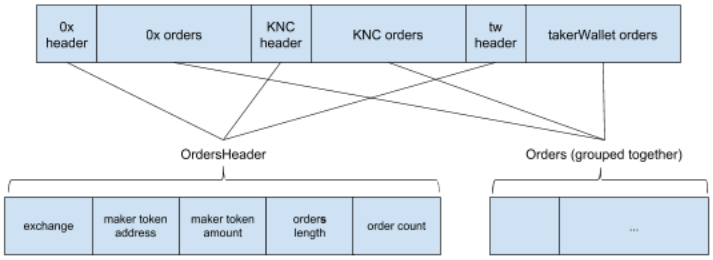

Figure 4: OrderData (input)

The data header for each exchange is formalized as follows:

Header Item	Type	Description
Exchange	uint8	Enumerated exchange wrapper identifier
Order Count	uint8	The number of orders for each exchange
Maker Token Address	address	Address of token being exchanged for components for a particular wrapper, must be consistent with makerTokenAddress of the issuance order
Maker Token Amount	uint256	Amount required to fulfill all of the orders in order body
Total Orders Length	uint256	Length in bytes of Order Body
Order Body	bytes	Orders concatenated in byte format

The data issuance order flow is formalized as follows:

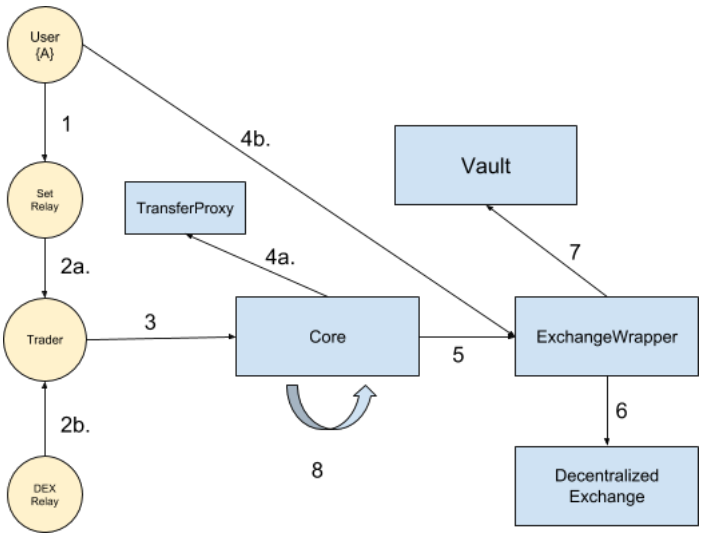

Figure 5: Issuance Order Flow

1. User (issuance order maker) desires to issue a Set ({A,B,C}) composed of {1x A, 1x B, 1x C}. User creates an issuance order, specifying the address of {A,B,C}, quantity to issue, maker token, component addresses needed, component quantities needed, and maker token amount willing to pay. User then submits the order to a Relayer that broadcasts this order on their orderbook.

   As an example, User wants to obtain 1 {A,B,C}. User owns Tokens A, but not B and C. The User is willing to pay 10 WETH for the remaining components. User creates an issuance order specifying address of {A,B,C}, 1 for the quantity, WETH as the maker token, 10 for the maker token quantity, B and C as the component addresses, and 1 and 1 as the component quantities.

2. Trader then looks at a separate decentralized exchange for exchange orders that fill 1x B and 1x C. Trader is incentivized to finds orders where the cost of B and C is less than 10 (as that is his profit opportunity). Trader finds an exchange order for 1x B that is 5 WETH and an order for 1x C that is 4 WETH. The sum of the cost is 9 WETH. The Trader stands to make 1 WETH.

3. With the B and C orders as well as the User's issuance order, Trader submits it to Core, which will execute the attached orders and issues {A,B,C}.

4. The User's maker token gets transferred into the ExchangeWrapper via the TransferProxy.

5. For each exchange order, the Core sends the order data to ExchangeWrapper.

6. The order gets parsed by the ExchangeWrapper and gets executed on the decentralized exchange with the ExchangeWrapper as the taker of the order. B and C tokens get transferred to the ExchangeWrapper.

7. Any remaining WETH gets sent to Trader, the submitter of the transaction. Each of the acquired component tokens gets transferred to the Vault and attributed to the User. Now that the User has all the required tokens to issue {A,B,C}, the issue function is called, transferring the User's A into the Vault and minting an {A,B,C}.

6.5 Redeem

Redemption is the process of converting a Set into its underlying component tokens. Redeeming tokens reduces the token supply of Set Tokens in the contract. It involves burning a Set Token and attributing the components to the end user. Users who wish to redeem a token can call Core's redeem or redeemAndWithdraw function, specifying the Set address and the quantity to redeem.

The redemption flow is formalized as follows:

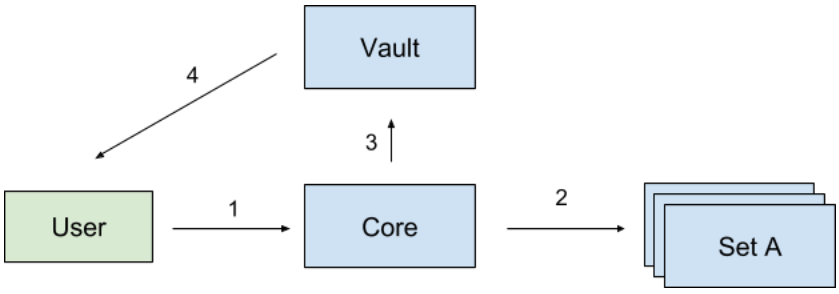

Figure 6: Redeem Flow

1. User calls the redeem function on Core, targeting Set A and specifying a redemption quantity. Core validates that the Set A is a known/approved Set.
2. Core queries Set A for the user's balance and checks the user has sufficient Set A tokens to redeem. If the user has sufficient balance, it queries the Set for its components, units, and natural unit. To prevent re-entrancy attacks, Core call Set A's burn function, decrementing the user's balance.
3. For each component, Core calls Vault to decrements the specified units attributed to the Set A. In each decrement, the Vault decrements the corresponding Set A's ownership of a token and increments the User's.
4. The user can subsequently call Core's withdraw function to retrieve the tokens from the Vault.

7 Rebalancing

7.1 Overview

Rebalancing is the process of realigning the weights of a portfolio, which involves periodically purchasing and selling assets to maintain a certain asset allocation. Typically, users actively rebalance their portfolios by reviewing their holdings, calculating the optimal allocation, and then trading their excess tokens for the missing tokens needed to create the new portfolio. In the traditional finance world where users purchase Index Funds or ETFs, fund managers (the counterparty) would perform the portfolio composition selection and rebalancing execution on behalf of users.

We introduce Rebalancing Set Tokens ("Rebalancing Set(s)"), a Set Token composed of another Set Token, called the currentSet. Using logic added to Rebalancing Sets, the currentSet can be transitioned to a new Set Token utilizing an oracle and a preconfigured rebalancing execution algorithm. Rebalancing Sets take a novel approach to rebalancing execution which completely removes counterparty risk. Rebalancing Sets can be configured to retrieve inputs regarding the most up to date Set Token through trusted or trustless sources.

Rebalancing Set Tokens can be used for many use cases including:

- **Tracking the Market Index**: Users who own Rebalancing Sets can passively track the market index by owning a single ERC20 token. Once users have acquired a Rebalancing Set, future rebalances execute behind the scenes, and they do not need to perform any additional transactions, mimicking the user experience of traditional ETF and Index Funds.
- **Delegating Portfolio Reallocation**: Users can create a Rebalancing Set that represents their portfolio and delegate future reallocations to a trusted third party or to the wisdom of the crowds.

- **Tracking an Influencer**: Users can own Rebalancing Sets that track the portfolios of influencers, VCs, or celebrities.

Users can generate Rebalancing Sets by acquiring the currentSet defined on the Rebalancing Set and calling issue. Managers can then propose changes to another Set Token. During the proposal period, users are free to maintain their Rebalancing Set if they agree with the change or redeem their Sets for the collateral backing if they disagree. Rebalancing execution is performed through a Dutch Auction, in which the outdated component Set Token is auctioned for the proposed Set Token. After this process, the user's Rebalancing Set Tokens are backed by the new Set Token at the conversion rate converged upon by the auction.

In a Rebalancing Set system, the participants include:

- **Users**: Rebalancing Sets enables users to get exposure to an automatically balanced crypto portfolio by simply purchasing a share of a Rebalancing Set on an exchange or by minting a Rebalancing Set themselves. Only if the user decides to opt out, does he or she need to get involved.
- **Oracle**: Each Rebalancing Set has a oracle that helps determine the updated Set to rebalance into. Initial implementations of a Rebalancing Set will have a single-partied Oracle, which effectively acts as the decision-maker. In the future, more trustless versions of Oracles can be introduced such as Token Curated Registries or on-chain Oracle solutions.
- **Rebalance Bidders**: During a Rebalancing Set's rebalancing period, profit-driven traders can help perform the rebalance by injecting the missing components of the new Set Token in exchange for the ejected components of the old Set Token.

7.2 Rebalancing Lifecycle

The Rebalancing Set life cycle has three states: Default, Proposal, and Rebalance. Each state enables functions that facilitate rebalancing from one Set Token to another. The Set can only move one direction through the cycle; there is no way to reverse or fast forward states.

7.2.1 Default

The Default state of a Rebalancing Set is the dormant stage in between rebalances. The Rebalancing Set Token will be in this state the majority of the time. In this stage, a Rebalancing Set Token operates no differently from a vanilla Set Token. Anyone can issue and redeem Rebalancing Set Tokens by calling the issue and redeem functions in Core. Rebalancing Sets are initialized in this state.

7.2.2 Proposal

A Rebalancing Set moves from the Default to Proposal when the Rebalancing Set Token manager submits a rebalance proposal. The proposal includes data about the updated Set and auction execution parameters. The proposal parameters include:

- **rebalancingSet**: Set that is being rebalanced in to, becomes new currentSet at end of Rebalancing period.
- **auctionPriceDivisor**: The granularity with which the price curve steps forward, higher number means more precision.
- **auctionStartPrice**: The price the auction starts at.
- **auctionLibrary**: Address of the library to be used for pricing.
- **curveCoefficient**: The slope or convexity of the auction curve.

During the Proposal period, token owners can review the proposal and opt out of the rebalance if they disagree with the terms. The length of the proposal period is defined upon contract instantiation. All the basic Set Token functions including issue and redeem remain available to all Rebalancing Set holders. While still in the Proposal state, a rebalance can also be re-proposed in the circumstance that there is component price volatility or an error is made. When the proposal period duration has elapsed, the Rebalancing Set can be transitioned from the Proposal state into the Rebalance state.

7.2.3 Rebalance

During the Rebalance state, the Rebalancing Set facilitates the transformation of the obsolete Set into the proposed Set in a series of steps. First, the obsolete Sets are redeemed for their components. Then, the Rebalancing Set runs a Dutch Auction to facilitate the exchange of obsolete components for the required new components (see next section for more detail). Finally, the updated Sets are issued using the new components and the Rebalancing Set moves to the Default stage. During this period, issuance and redemption are disabled.

7.3 Rebalancing Dutch Auction

The Rebalancing Set initiates a public auction where outdated components are exchanged for the latest components. The Rebalance Set uses an auction structure where the lowest price ratio (new components/old components) is initially offered, and the price ratio declines as a function of a predefined proposal price curve.

At any point, a user ("Bidder") can submit a transaction to execute a trade ("Bid") to exchange the new components for the old at the current price ratio. As the price ratio increases, the price becomes more and more favorable for Bidders. The auction is concluded when all the old components have been replaced with the new components.

7.3.1 Bidding

In the dutch auction, bidding is the act of the executing a rebalance. When users bid, they inject new Set components to facilitate a rebalance and in return immediately receive the old Set components. Bidders are incentivized to transact when there is a spread between the cost of acquiring the injected components and the received components. Bidding is open to anyone, and multiple parties can transact as long as the dutch auction is running.

Bidding has been designed to minimize the capital required to participate and price risk for bidders. We achieve this by only requiring injections of constituent tokens that are involved in the rebalance and providing access to returned tokens as soon as the bid is recorded. This way, the bidder only needs to acquire the components required by the system to undergo the rebalance.

7.3.2 Price Determination

Price is determined when the bidder's transaction is mined and is calculated as a function of the auction parameters set out by the manager, and the time since the auction began. Each rebalance references an Auction Library that is used to calculate the price of the bid. Auction Libraries are separate contracts that have their own price curve (linear, log, cubic, etc.). Using these inputs, managers have the ability to tune parameters to increase likelihood of getting a fair price and bidders have the transparency of knowing how the price will change over time.

8 Governance

Initially, governance will managed by the team using a multi-signature contract. Our intention is to create a system that is as decentralized and trustless as possible. See below for the limited capabilities of the governors:

- Add and remove ExchangeWrappers and Factories
- Put Sets in a redeem-only mode and disable new issuance Orders from being filled.
- Put the protocol in an emergency state where speed bumps for critical functions like Redeem and Withdraw are enforced in the case a major security vulnerability is found.

Over time, Set expects to move these rights to a DAO controlled by the community or utilizing a governance framework like Commonwealth or Aragon.

9 Future Work

Because we are still very early days in development of Set, there are still many outstanding areas and features that we plan on developing. We plan on building Set to meet the needs of the broadest possible community. Some of these include:

9.1 Alternative Asset Support

- **ERC-721 Support**: Sets that represent fractional ownership of a collection of ERC721 Assets[7]. Some of these use cases include fractional ownership of non-fungible tokenized assets such as art, title deeds, and sports teams.

  Implementing issuances of ERC721 Sets is trivial, but redemption logic is a bit more complicated. Some governance questions that arise include: 1) How do the token-holders decide when to redeem their shares for non-fungibles? 2) What happens when there is disagreement between shareholders (e.g. divorce)? 3) In the case that shareholders decide to sell, how are assets distributed? Do they get auctioned for fungible assets first?

9.2 Alternative Vault Implementations

- **Interest-Bearing Vault**: Vault where components can be enabled for lending in fully/non-fully collateralized basis (e.g. Marble flash loans, Compound money markets, Dharma). This allows components to passively generate income for Set token holders.

  Some technical challenges include the attribution of interest to particular Sets and thus Set holders and integrating with various interest-bearing protocols/creating our own. Some governance questions include: 1) What Sets are redeemed in the case that borrowers have defaulted on their components? 2) How does the protocol handle situations where collateral has been seized?

- **Staking Vault**: Vault where components have been enabled by Set Tokens to stake into staking protocols (e.g. integration with Vest, Casper, Livepeer, TrueBit). The technical challenges of the Staking Vault are similar to that of the Interest-Bearing Vault.

- **Governance-Enabled Vault**: Vault that stores governance tokens where voting rights have been delegated to a DAO or Set token manager. Examples of protocols where tokens can be used for governance include 0x Project, Commonwealth, and Kleros.

10 Summary

We introduce the idea of a Set, which is a single token that represents a basket of tokens. Sets are a powerful tool for allowing us to hide away the details of underlying tokens and focus on higher-level concepts. It represents a new asset class that allows more efficient transactions, is fully collateralized, and is trustless. Sets serve as a fundamental building block for composing more complex financial instruments. Sets are a primitive and serve as a powerful tool that allows developers and product-creators to build increasingly complex decentralized applications that are user-friendly and intuitive. There is a limitless number of Set use cases.

11 References

1. Coinmarketcap. https://coinmarketcap.com/ (Accessed November 2017)
2. 0x project. https://0xproject.com/pdfs/0x_white_paper.pdf
3. dy/dx. https://dydx.exchange/
4. dharma. https://dharma.io/whitepaper/
5. ERC20 https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
6. Ethereum Smart Contract. http://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html
7. ERC721. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md

12 Appendix

There is a limitless number of use cases for Sets including:

- **Carbon Neutral Token**: A Set that is composed of a barrel of oil and carbon tax credits.
- **Bundling for Gifts**: A Set composed of a bundle of tokens to gift to their friends and family.
- **Healthcare Identity**: A Set that aggregates one's healthcare information (represented as individual tokens) for transfer to healthcare providers when requested.
- **Diversified Portfolio of Bonds**: A Set that aggregates bonds of various interest rates and expiration dates for investment.
- **Student Loan Token**: A Set composed of tokens that can restrictively be redeemed for tuition and housing.
- **All-Inclusive Vacation Package**: A Set representing a tourist package which aggregates flights, hotel stay, meal vouchers, tours, and local transportation.
- **Loyalty Programs**: A Set that represents a user's loyalty rewards programs including airlines, hotels, car rental programs.