dYdX: A Standard for Decentralized Margin Trading and Derivatives

Antonio Juliano

September 25, 2017 *[Updated August 6, 2018]*

Abstract

We present a set of protocols that allow several types of financial products to be created, issued, and traded for any pair of underlying ERC20 tokens. Our approach uses off-chain order books with on-chain settlement to allow creation of efficient markets. All described protocols are fair and trustless, creating truly open markets that are not governed by a central authority. The protocols are extensible by anyone, requiring no special permissions to be used with other smart contracts.

Contents

1 Introduction

The rise of blockchains has enabled anyone to own and transfer assets across an open network without needing to trust any external parties. Unlike existing financial architecture, blockchains are freely and equally available worldwide. This has led to a large and rapidly increasing number of digital assets existing on the blockchain. Many centralized and decentralized platforms designed to facilitate the efficient exchange of these assets already exist, and more are in development. Such platforms allow investors to take long positions in various assets. However, it is currently very difficult or impossible to take more complex financial positions.

dYdX allows creation of entirely new asset classes which derive their value from underlying blockchain-based assets. Financial products such as derivatives and margin trades allow investors to achieve superior risk management with their portfolios, and open up new avenues for speculation. They also increase market efficiency for the underlying asset by aiding in price discovery and allowing individuals to express more complex opinions on price and volatility. dYdX provides advantages over traditional financial products by eliminating the need for a regulated central clearing house, providing global and equal access, and allowing users full control of their funds at all times.

The size of the derivatives market on existing financial infrastructure far outstrips the market size of any other type of financial asset. It is roughly estimated to be over $1.2 quadrillion[1], or more than 10 times the total world GDP. We believe that as decentralized platforms mature and start to offer significant advantages over traditional financial systems, an ever increasing number of traditional assets will start to be listed on the blockchain.

dYdX will offer a number of decentralized protocols implementing various types of crypto-asset financial products. These protocols are comprised of open source Ethereum Smart Contracts and standards.

[1] Investopedia. *How big is the derivatives market?*.
http://www.investopedia.com/ask/answers/052715/how-big-derivatives-market.asp

2 Existing Work

There are few existing decentralized protocols that support derivatives or margin trading and none that have any significant usage. Centralized exchanges also fail to offer adequate financial products on decentralized assets. Consequently, it's very difficult to take short or more complex financial positions on the bulk of today's decentralized assets.

In order for a decentralized derivatives or margin trading protocol to operate, there needs to be a way to trustlessly exchange assets, as well as determine the price at which assets will be exchanged. A decentralized exchange protocol is one that facilitates the trustless exchange of one token for another at prices dictated by the market. dYdX can work with any standard Ethereum-based decentralized exchange. Initially, dYdX will use the 0x protocol[2] to enable token exchange at rates supplied by users of the protocol.

Several types of decentralized exchanges have been proposed: on-chain order books, automated market makers, state channels, and a hybrid off-chain order book approach. The 0x whitepaper offers an in-depth discussion of the tradeoffs between these models[3]. We chose to base dYdX on the hybrid approach pioneered by 0x, as we believe it allows creation of the most efficient markets. This allows market makers to sign and transmit orders on an off-blockchain platform, with the blockchain only used for settlement.

One previous attempt at decentralized derivatives, Velocity[4], proposed using an oracle based approach to feed the exchange rates of asset pairs to a smart contract responsible for operation of options contracts. The contract would then use this price information to create and exercise options. Using such an oracle based approach has several significant drawbacks. The limitations on frequency, latency, and cost of price updates due to the nature of blockchains makes it impossible to create markets as efficient as those built on traditional centralized exchanges. Using an oracle also adds a great deal of centralization to any protocol, as some central parties have full control over setting the price. Worse, if those central parties were also trading on the protocol, they would have a huge economic incentive to manipulate prices in their favor.

dYdX protocols allow trade of financial products at any price agreed upon by two parties. This means that there is no need for the contracts to be aware of the market price. Traders provide orders of their choosing, which are then used to execute the exchange. It is in the economic interest of traders to choose orders with the best prices. This best price is dictated by the market, and no orders with better prices will exist.

---

[2] Will Warren, Amir Bandeali. *0x White Paper*. https://0xproject.com/pdfs/0x_white_paper.pdf

[3] Will Warren, Amir Bandeali. *0x White Paper*. "Existing Work". https://0xproject.com/pdfs/0x_white_paper.pdf

[4] Shayan Eskandari, Jeremy Clark, Vignesh Sundaresan, Moe Adham. *On the feasibility of decentralized derivatives markets*. https://users.encs.concordia.ca/~clark/papers/2017_wtsc.pdf

3 Protocols

dYdX consists of a number of protocols specifying the operation and execution of different types of financial products. We plan to prioritize the development of the most popular and widely used types. Below we outline our implementation of protocols for options and margin trades. We plan to develop protocols for additional types of financial products in the future.

3.1 Margin Trading Protocol

3.1.1 Description

In a margin trade, a trader borrows an asset and immediately trades it for another asset. The asset must be repaid to the lender, usually along with interest, at a later date. Margin trading includes both short sells and leveraged longs.

In a short sell an investor borrows an asset and sells it for the quote currency. The investor makes money if the price of the asset decreases, since rebuying the asset to repay the lender costs less than the original sell-price.. The investor loses money if the price of the asset increases, since rebuying the asset to repay the lender costs more than the original sell-price. The lender makes money from the interest paid by the trader.

In a leveraged long an investor borrows the quote currency and uses it to buy an asset. The investor makes money if the price of the asset increases, and loses money if it decreases. Gains or losses from the position are equal to the change in price of the underlying asset multiplied by the leverage ratio, which is the ratio of the sum of the borrowed amount plus the amount paid by the trader to the amount paid by the trader.

3.1.2 Use Cases

Short sells are used to enable investors to profit from an asset which decreases in price. Short sells can be used for both speculation and hedging. Investors can use a short sell for speculation when they believe the price of an asset will go down. Short sells can be used to hedge existing positions by shorting a correlated asset.

Leveraged longs are used to multiply gains when an asset increases in price. Leveraged longs can be used for speculation, as they allow traders to achieve larger gains with less capital. Investors can use leveraged longs for more efficient capital allocation, as less capital is required to achieve the same results for each investment.

Lending assets for margin positions can provide the lenders with interest from the loan.

3.1.3 Overview

The dYdX Margin Trading protocol uses one main Ethereum Smart Contract to facilitate decentralized margin trading of ERC20 tokens. Lenders can offer loans for margin trades by signing a message containing information about the loan such as the amount, tokens involved, and interest rate. These loan offers can be transmitted and listed on off-blockchain platforms.

A trader opens a margin position by sending a transaction to the dYdX margin smart contract containing a loan offer, a buy order for the borrowed token, and the amount to borrow. Upon receiving this transaction, the smart contract transfers the margin deposit from the trader to itself, and then uses an external decentralized exchange such as 0x to sell the loaned token using the specified buy order. The smart contract holds onto the deposit and token resulting from the sale of the loaned token for the life of the position.

The position is closed when the trader sends a transaction to the smart contract containing a sell order offering to sell the amount of token owed to the lender for an amount less than or equal to the amount locked in the position. Upon receiving this transaction, the contract uses an external decentralized exchange to execute the trade between the order maker and itself. After, the contract sends the owed amount of the loaned token to the lender. The amount owed to the lender includes the interest fee. The trader is sent all of the leftover token, which is equal to $deposit + profit$. Note the profit could be negative if the price moved against the position.

The loan for a margin trade can also be called in by the lender when the price has moved against the position. Once the loan is called in, the trader has a specified amount of time to close the position. The trader can also allow other contracts to trustlessly and automatically close the position on their behalf using mechanisms such as a dutch auction.

The margin trading protocol can be used for both short selling and leveraged long trading by simply switching which token is borrowed (referred to as the *owed token*) with the one that is held in the position (referred to as the *held token*)[5]. The protocol allows the margin deposit to be paid in either *owed token* or *held token*. If the deposit is paid in *owed token* it is sold along with the *owed token* borrowed from the lender, so that only *held token* is held in the position. Similarly, the payout to the trader from closing can be in either *owed token* or *held token*. If the payout is in *owed token*, all *held token* in the position is sold for *owed token* and whatever is leftover after paying the lender is paid out to the trader.

When used for short selling, the trader will borrow *base token* from the lender which will be sold for *quote token*, and put up a margin deposit in *quote token*. Only *quote token* is held in the position. When the position is closed *base token* will be bought and paid back to the lender, and the trader will be paid out in *quote token*.

[5] The concept of *owed token* and *held token* should not be confused with *base token* and *quote token.* Depending on whether it is a short sell or leveraged long position, *base* or *quote token* can be the *owed* or *held token* in the position. This section aims to articulate the relatedness of the two positions from an implementation point of view.

When used to take a leveraged long position, the trader will borrow *quote token* from the lender, and put up a margin deposit in *quote token*. Both the *quote token* borrowed from the lender, as well as the *quote token* put up as margin deposit are then sold for *base token*. Only *base token* is held in the position. When the position is closed *all* of the *base token* is sold for *quote token*. *quote token* is paid back to the lender, and the trader is again paid out in *quote token*.

3.1.4 Implementation

3.1.4.1 Contracts

For margin trading, there are three contracts used: the *Margin* contract, the *Proxy* contract, and the *Vault* contract.

The *Proxy* is used to transfer user funds. Users set token allowances on the *Proxy* which authorizes it to transfer funds on their behalf.

The *Margin* contract offers functionality to enable margin trading. It contains all the business logic and public functions. It also contains the state where positions are stored. The *Margin* contract is designed so existing positions cannot be modified by any external party (see the governance section).

The *Vault* contract holds all the funds locked up in positions. It exposes a simple interface which the *Margin* contract is authorized to use.

3.1.4.2 Offering Message

The first ingredient to a margin trade is a lender who holds the *owed token*, and wants to lend it out for a given deposit and interest rate. The lender prepares and cryptographically signs a message with the following information:

Name	Type	Description
owedToken	address	Address of *owed token* - the token borrowed from and owed to the lender
heldToken	address	Address of *held token* - the token held in escrow by the position
payer	address	Address that supplies the funds for the loan. If this is different than *signer* it is assumed to be a smart contract and its consent is gotten through an interface
signer	address	Address that cryptographically signs the loan offering
owner	address	Address that will own the loan after it is taken. All payouts will

		go to this address
taker	address *(optional)*	If set, only this address will be able to take the loan
feeRecipient	address *(optional)*	Address to receive relayer fees associated with this offering
lenderFeeToken	address *(optional)*	Address of the token to charge the lender fee in
takerFeeToken	address *(optional)*	Address of the token to charge the taker fee in
maxAmount	uint256	The maximum amount of the loan offering. Denominated in units of *owed token*
minAmount	uint256	The minimum takeable amount of the loan offering. Denominated in units of *owed token*
minHeldToken	uint256	The minimum amount of *held token* locked in the position after the deposit and sell (based on *maxAmount*)
lenderFee	uint256 *(optional)*	Amount of *lenderFeeToken* to charge the lender (based on *maxAmount*)
takerFee	uint256 *(optional)*	Amount of *takerFeeToken* to charge the taker (based on *maxAmount*)
interestRate	uint32	The interest rate (continuously compounded, represented as annual nominal percentage with up-to 6 decimal places)
interestPeriod	uint32 *(optional)*	The interest rate update period. Interest fee will increase once per period
expirationTimestamp	uint32	The timestamp (in seconds since unix epoch) at which the offering expires
callTimeLimit	uint32	The minimum amount of time (in seconds) that the position must be closed after being margin-called by the lender
maxDuration	uint256	The maximum duration (in seconds) of the loan. Relative to when a position is opened

This message can then be broadcast off-blockchain between counterparties. It is a binding agreement to commit to the loan if a trader desires. The protocol is agnostic to the medium of exchange used to relay these signed messages. It is expected that these offers will be listed on centralized platforms referred to as

relayers and will compete on interest rate and terms. Larger OTC trades can be agreed upon through traditional means, then made formally-binding using the protocol.

3.1.4.3 Buyer

The second ingredient to a margin trade is a buy order which can be filled as part of the margin trade.Like the loan offering, the buy order can be transmitted through any means. The buyer is in no way involved in the loan or margin trade. This order can be for any price, and must be selected by the trader. The only prerequisite is the order must be for at least as much *owed token* as the trader is selling as part of the margin trade. It is in the trader's economic interest to select the buy order with the best price.

dYdX allows any standard buy/sell decentralized exchange to be used. This is done by wrapping external decentralized exchange smart contracts in another contract that provides standard interface to *Margin*. The wrapping contract is known as an *ExchangeWrapper*. The *ExchangeWrapper* is specified by the trader for each margin trade and requires no special permissions. This means anyone can write, deploy, and use an *ExchangeWrapper* for any decentralized exchange. dYdX has implemented the first *ExchangeWrapper* which wraps the 0x Exchange Contract, and allows any 0x order to be used to open a dYdX position.

3.1.4.4 Position Opening

To open a position, a trader sends a transaction to the *Margin* smart contract containing:

- The signed loan offering
- The buy order offering to buy *owed token* for *held token*
- The address of the *ExchangeWrapper* to be used with the buy order
- The amount of *owed token* the trader wishes to borrow
- A boolean indicating whether the trader wishes to post margin deposit in *held token* or *owed token*
- The amount of token the trader wishes to put up as a deposit
- The address that will own the position after it is opened

When the contract receives the transaction the following happens:

1. The signature and inputs on the loan message are verified
2. *Margin* calls into *Proxy* to transfer the offered deposit in either *held token* or *owed token* from the trader to *Vault* (if depositing in *held token*) or to the *ExchangeWrapper* (if depositing in *owed token*)
3. *Margin* calls into *Proxy* to transfer the requested amount of the *owed token* from the lender to the *ExchangeWrapper*

4. *Margin* records that the requested amount of the loan has been used, and saves it in a mapping. This is used to keep track of the amount remaining in the loan offer and protect against replay attacks using the signed loan message[6]
5. *Margin* calls into the *ExchangeWrapper* to exchange the *owed token* for the amount of *held token* offered by the buy order. The buyer is the maker in this trade and the *ExchangeWrapper* is the taker. The exchange contract (e.g. the 0x Exchange Contract if using 0x) will verify the inputs and signature on the supplied buy order and execute the trade
6. *Margin* calls *Proxy* to transfer the *held token* received from the sell from the *ExchangeWrapper* to *Vault*. The *held token* remains locked in *Vault* for the duration of the position
7. The details of the position are stored in the contract, mapped by a unique public identifier for the position. This identifier is used by the trader and/or lender to interact with the position at a later date

All steps happen atomically, meaning that they all succeed or all fail together. At the end, the *Vault* contract ends up with an amount of *held token* for the position. If the margin deposit was put up in *held token*, this amount is equal to the deposit put up by the trader plus the *held token* resulting from the sale of the *owed token*. If the margin deposit was in *owed token*, the amount is equal to the *held token* resulting from the sale of both the borrowed *owed token* as well as the *owed token* put up as margin deposit. *Vault* holds onto these funds until the position is closed.

3.1.4.5 Closing

The trader can decide to close any portion of the position at any time by presenting the *Margin* contract with a sell order offering to sell greater than or equal to the amount of *owed token* owed to the lender (including interest fee) for an amount of *held token*. This sell order can be for any price such that there is enough *held token* in the position (prorated by the portion of the position being closed) to pay for it. However it is in the trader's economic interest to select an order with the lowest price.

When *Margin* receives this transaction, the following happens:

1. The total amount (in *owed token*) owed to the lender at this point in time is calculated using continuously compounded interest
2. *Margin* calls into the *ExchangeWrapper* to execute the trade of *held token* for the amount of *owed token* owed (if paying out in *held token*), or to trade all of the *held token* in the position for *owed token* (if paying out in *owed token*). After the trade, *Vault* holds the owed amount of *owed token* and an amount of either *held token* or *owed token* equal to $deposit + trader\ profit$ (profit could be negative) for the position
3. *Margin* calls into *Proxy* to transfer the owed amount of *owed token* from the *ExchangeWrapper* to the lender
4. *Margin* sends either *held token* or *owed token* equal to $deposit + profit$ to the trader

[6] Will Warren, Amir Bandeali. *0x White Paper*. "Fills & Partial Fills". https://0xproject.com/pdfs/0x_white_paper.pdf

5. *Margin* deletes the position from its storage if its value is now zero, or reduces its amount by the amount that was closed

At the end of the margin trade, the trader ends up with the *profit* amount, denominated in either *held token* or *owed token*. The lender makes the amount of interest fee in *owed token*. The *Margin* / *Vault* contracts end up net neutral as desired.

3.1.4.6 Calling

The other way a margin trade can be settled is by the lender or another party authorized by the lender calling in the loan from the trader. It is done by the lender or authorized party sending the *Margin* contract a transaction indicating they are calling in the loan, along with an amount of *held token* that must be deposited into the position by the trader to cancel the margin call. After this transaction the trader has the amount of time originally specified in the loan (call time limit) to either pay back the loan, or put up additional *held token* as deposit. The trader uses the same process described above in the closing section to close the position. If the trader fails to close the position or put up the required additional deposit, the lender is entitled to the entire *held token* balance locked in the position.

It is in the lender's interest to call in the position when the price of *owed token* relative to *held token* rises to the point that the *held token* locked in the position is almost not enough to buy back the owed amount of *owed token*. This means the lender or authorized party needs to be watching the price and be ready to call in the position on an upward price movement. The authorized party would most likely be a relayer or service that watched the price and was always ready to programmatically call in loans on price movements. The approach of authorizing a trusted party is functionally equivalent to using a centralized oracle to determine when margin calls should occur, however is more efficient as gas does not need to be paid for constant price updates and the price can be effectively watched in real-time rather than once per block.

This approach also requires that the trader is always online and able to send a transaction to close the position before the call time limit, or risk forfeiting the entire position balance. To protect traders from always having to be online, traders can optionally opt-in to an external contract that can automatically and trustlessly close the position on their behalf.

The automatic closing contract works by running a dutch auction offering to buy back the amount of *owed token* owed to the lender for an amount of *held token* that starts at 0 and linearly increases to the total amount of *held token* in the position over the call time limit. Any excess *held token* is given to the trader. Anyone can bid on the auction, or use an existing decentralized exchange order to buy the *owed token* and keep the spread. The moment the auction price crosses the market price, there will exist an incentive for everyone to bid on the auction, causing the trader to get paid out at market price.

3.1.5 Risks

One risk for the trader is that the lender calls in the loan before the trader wishes to close the position even when enough deposit is posted. Current non-blockchain related financial systems use a reputation system to identify optimal lenders that will not call in the loan prematurely. Such a reputation system for dYdX could exist entirely separately to the base protocol, as traders would prefer loan offers from lenders with higher ratings and would price this into their decision on whether or not to take a loan. Another solution is to use an authorized party that is mutually trusted by both the trader and lender to margin call the position. In the future, decentralized price oracles could also be used.

The risk for the lender (besides the economic risk of holding the *owed token*) is that the price of the *owed token* relative to the *held token* rises so rapidly that the loan is not able to be margin-called before the amount of *held token* locked in the position is no longer enough to buy back the owed amount of *owed token*. In this case the lender would still receive the entire amount of *held token* locked in the position, but would have been better off just holding the *owed token*. This risk for the lender can be mitigated by setting a high enough deposit, low enough call in time, and by using an efficient margin-calling mechanism (likely through off-blockchain monitoring).

3.2 Options Protocol

3.2.1 Description

In an option, a holder of an asset sells the right to buy or sell that asset at a specified strike price and future date[7]. An option to buy an asset is referred to as a call, and an option to sell an asset is called a put. The seller of the option (the writer) collects a premium upon sale, but is also bound to buy or sell the asset at the agreed upon price and date if the holder of the option desires. A covered option indicates that the underlying asset is put up as collateral, so it is guaranteed to be able to be collected at a future date. The option can itself be traded on the open market. We describe an implementation of an American covered option, or one which can be exercised at any time before the expiration date.

3.2.2 Use Cases

Options enable numerous trading strategies that can be designed for speculation or risk management.

Options can be used to provide additional leverage in speculation. For example suppose the price of AAPL is $100, and an investor who has $1000 to invest believes it will go up. The investor could buy 10 shares at $100, and if the price rises to $110, selling would yield a $100 or 10% profit. Suppose instead that the investor had purchased call options with a $100 strike and $2 premium. The investor could afford 500 of these options with $1000. If the price again rose to $110, the investor could exercise the options to buy at $100, and then immediately sell at $110 for a $10 profit per option. Since the investor had paid $2 for each option, a profit of $8 per option would have been made. This means the investor's profit would have been $8 * 500 = $4,000 or a 400% return. This shows how with the same amount of capital investors can achieve much larger returns using options than by simply holding the asset.

Options can also be used to hedge or reduce risk in an investment. Imagine an investor is long 100 share of AAPL, which is again trading at $100. The investor could purchase a put option with $90 strike for a $2 premium. Such an option would ensure that for only a 2% fee, during the lifetime of the option the investor could not lose more than 10% on the investment.

Options also enable more advanced trading strategies such as straddles, strangles, collars, and many more. Among other things, such strategies can lock in a price, profit from volatility in any direction, or profit from price stability in an asset.

3.2.3 Overview

The dYdX option protocol uses one Ethereum Smart Contract per type of option. A type refers to a given set of input parameters including the *base token*, *quote token*, strike price, and expiration date. *base token* refers to the asset the option is for and *quote token* refers to the token in which the premium and strike

[7] Investopedia. *Options Basics: What are Options?*. http://www.investopedia.com/university/options/option.asp

price are denominated[8]. Each option contract is able to issue new options of its type at any time before the option expiration date. The contracts can act as either a put or a call option by simply switching the *base token and quote token* and inversing the strike price.

Writers of the option list offers for a specified lot size and premium on an off blockchain platform. Buyers can buy options from a writer by sending a transaction containing a write offer to the smart contract. After receiving such a transaction, the smart contract transfers the premium in *quote token* to the writer, and the offered amount of *base token* to itself. The buyer is issued options which can be transferred and traded as any other ERC20 token. The smart contract holds on to the *base token* until the option is either exercised or expired.

Any holder of the option can choose to exercise at any time before the expiration date. Upon exercise, the option holder pays $strike\ price \times (\#\ options)$ of *quote token* to the smart contract and is sent $\#\ options$ of *base token* from the smart contract. The *quote token* paid to the contract is distributed to the writer or writers of the option. After the option expires, all writers can withdraw *base token* from the smart contract corresponding to $\frac{Options\ Written}{Total\ options\ Written} \times (total\ tokens\ held)$.

3.2.4 Implementation

3.2.4.1 Contracts

We use three types of smart contracts to allow the issuance and functionality of options: the *Creator*, *Proxy*, and *CoveredOption* contracts.

The *Creator* is responsible for creating all *CoveredOption* contracts. Anyone can create a new type of *CoveredOption* by providing the the following specifications:

- The address of the ERC20 token the option is for (referred to as *base token*)
- The address of the ERC20 token the strike price and premium are to be paid in (referred to as *quote token*)
- The strike price (broken into two parts to form an exchange rate between *base token and quote token*)
- The expiration date

Creating a new type of *CoveredOption* only opens it up for sale, and does not issue any options. There can exist only one *CoveredOption* for each combination of input parameters.

The *Proxy* is responsible for transferring user tokens between accounts. Users use the ERC20 allowance functionality to authorize the *Proxy* to move their tokens. Each new *CoveredOption* is authorized to use the *Proxy* to transfer user funds when it is created by the *Creator*.

[8] Investopedia. *Base Currency*. http://www.investopedia.com/terms/b/basecurrency.asp

The *CoveredOption* contract represents a specific type of covered option. Each one implements the ERC20 interface to allow shares of the option to be traded and transferred after issuance. This means every option can be publicly traded on an exchange as any other ERC20 token.

3.2.4.2 Issuance

CoveredOption uses the exchange functionality of the 0x Protocol to facilitate issuance of new options. Options can be issued anytime before the expiration date of the option. In order to issue new options, the writer broadcasts a signed message in the 0x message format specifying the following information:

- The address of the writer
- The address of the fee recipient
- The amount of *base token* the writer is offering
- The amount of *quote token* to be paid as a premium to the writer upon purchase
- The expiration time for the sale of this option
- The address of the *CoveredOption* contract for the option they want to write. This address is specified in the taker field of the message, so only the *CoveredOption* contract can take the trade

The writer must have at least as much *base token* as offered, and must set allowance on the *Proxy* contract. Buyers can buy less than the amount of options offered by the writer. In 0x terminology, the writer will be the maker of the trade, and the *CoveredOption* contract will be the taker of the trade. The message can be published in any channel, but is a binding agreement to offer the specified sale. Relayers can then list these option sale offers on an option issuance order book (much the same as relayers in the 0x protocol).

When a buyer wants to purchase an option, they send a transaction to the *CoveredOption* contract that includes the message signed and broadcast by the writer, and the amount of options they wish to buy. Options are issued on a 1:1 ratio with the amount of *base token* deposited by the writer. Once the *CoveredOption* contract receives this transaction it does the following:

1. Validates the expiration date of the option has not yet passed
2. Calls into the *Proxy* to transfer the appropriate amount of *quote token* from the buyer to the *CoveredOption* contract itself. This is the premium that is being paid for the option.
3. Call the *0x Exchange Contract* to exchange the *quote token* which was just taken from the buyer with the appropriate amount of *base token* from the writer. The *0x Exchange Contract* validates the the writer's signature, ensuring this offer is legitimate. The writer is the maker and the *CoveredOption* contract is the taker in this trade. After this, the writer ends up with the *quote token* premium, and the *CoveredOption* contract ends up with the offered amount of *base token*. The *CoveredOption* contract will hold the *base token* until the option is settled.
4. The *CoveredOption* contract records that the writer has deposited the amount of *base token*. This amount is used later in the case the option expires without being exercised.

5. The balance of the buyer is increased by the amount of options purchased. The buyer is now the holder of that amount of the options, and can now freely transfer and trade them as per the ERC20 standard.
6. If the amount of options available to be written was less than the amount desired by the buyer, the excess *quote token* left over after the trade is transferred back to the buyer.

All of the above steps happen atomically (i.e. they all happen, or none of them happen) in a single transaction.

3.2.4.3 Exercise

Before the option expires, any holder of the option can exercise any amount up to the number of options owned. This means the holder agrees to pay the strike price (globally specified on the *CoveredOption* during its creation), for every option exercised. It is only in the holder's economic interest to exercise the options if the market price for the *base token* is greater than the strike price of the option.

In order to exercise, the owner sends a transaction to the *CoveredOption* contract indicating how many options are to be exercised . Assuming the transaction is valid, the *CoveredOption* contract:

1. Calls into the *Proxy* to transfer $strike\ price \times \#\ options$ of *quote token* from the sender to the *CoveredOption* contract itself
2. Deducts balance from the owner
3. Sends the owner *base token* on a 1:1 basis with number of options exercised
4. Holds onto the *quote token*. The appropriate portion can later be withdrawn by each writer of the option

3.2.4.4 Withdrawal

After the option expires, any writer of the option can withdraw a proportion of both *base token* and *quote token* held by the *CoveredOption* contract corresponding to:

$$\frac{Options\ Written}{Total\ options\ Written} \times (total\ tokens\ held)$$

This is done by sending the *CoveredOption* contract a withdraw transaction, which causes the contract to send the writer their full balance of each token, and sets the writer's written balance to zero.

If an address is both the writer and holder of an equivalent number of options, it may at any time withdraw any amount of *base token* less than or equal to:

$$min(\#\ options\ written,\ \#\ options\ held)$$

Doing so will decrease both the address's balance and number of options written by the amount withdrawn. This is provided as a utility so a writer can always get the *base token* back, even before the option expires, by purchasing the desired number of options.

4 Governance

Governance will initially be handled by a multisig contract whose keys are held by reputable individuals with a vested interest in the success of dYdX. The powers of this contract will be limited to putting the dYdX protocol into a close-only mode, preventing the creation of any new positions. The contract will have no power to influence any open positions, nor will the contract be able to add new functionality to the protocol. A lack of centralized power is essential to the trustlessness of the protocol. The limited power to put the protocol into close-only mode is intended to be used only to protect would-be users in the event that a major security bug is found.

dYdX enables anyone to increase the functionality of the protocol by allowing users to specify their own smart contracts to help open, close, or manage positions. In this way, any upgrades are completely opt-in by users of the protocols themselves and can also be written by anyone, requiring no special permissions from the base protocol.

In this way, upgrades cannot be forced by the authors of the protocol. Therefore, a token is not currently needed for governance. In the future, to help promote common standards, dYdX will consider using a DAO to govern upgrades to the protocol, however no viable DAOs currently exist.

5 Summary

- dYdX
  - Decentralized protocols for peer-to-peer derivatives and margin trading
  - Built on Ethereum and 0x
  - Open-source and free to use
  - Efficient markets are enabled using off-chain 0x orders and economic incentives for price discovery
  - Modular, extensible smart contracts allow continuous opt-in upgrades
- dYdX Margin Trading Protocol
  - Can be used to profit on downward price movements, or increase leverage
  - Providing low risk fully collateralized loans for margin trades can provide interest fee on long positions
  - Anyone can margin trade or lend any ERC20 token
- dYdX Options Protocol
  - Can be used to reduce risk or speculate
  - Anyone can create, write, buy, or trade any option on any ERC20 token
  - Each option is represented by its own ERC20 token to allow easy trading

6 Acknowledgments

We would like to thank our mentors, advisors, and friends who have provided invaluable advice on dYdX. In particular, we would like to thank Olaf Carlson-Wee, Fred Ehrsam, Linda Xie, and Julian Borrey for reviewing and providing feedback on this work. We would also like to thank Albert Zhou for his help educating us about derivatives. dYdX would not have been possible without their and others' support.