2 Feb, 2022

1.0 Introduction to DeFi Fixed Income

The most popular collateralized lending/borrowing protocols today include Compound, Aave, and MakerDAO. These protocols offer a floating interest rate via a pool which users can supply or borrow from. While these protocols solve a variety of use cases, there is increasingly more demand both from lenders and borrowers for more predictable, fixed interest rates.

Furthermore, while there are numerous protocols that offer staking for yield on tokens, and numerous protocols that offer borrowing stablecoin, there are perhaps no protocols that currently alow borrowing of altcoins at fixed rates. This is important because the ability to borrow altcoins enables users to short the token (i.e. borrow and sell the token today for USD, and buy back in the future at a lower rate), and needs to be at fixed rates because a floating cost of borrow presents too much risk to traders, especially if there is a short squeeze. The option to short a security is vital to the overall health of the market, and there are perhaps no protocols available today that enable this function. Qoda aims to democratize this process, as the protocol that enables users easy access to hedging or shorting crypto exposure by allowing all DeFi participants equal opportunity access to altcoin lending/borrowing markets.

Lastly, in the long term, the path to growth and maturity of the DeFi fixed income space should begin with the mindset that interest rates as a product can be tradeable in an orderbook exchange type of environment. Furthermore, with accurate traded interest rate data across a variety of maturities, one can start to bootstrap the construction of a crypto Y*ield Curve*. As in traditional financial markets, the yield curve is the fundamental building block upon which more complex and customized fixed income products can be built. Much in the same way Uniswap functions as an on-chain oracle for real-time spot currency prices, Qoda can become the de facto on-chain oracle for crypto yield curve data.

With these goals in mind, we hope to achieve Qoda's mission: *To bring tradeable fixed income markets to the DeFi world.*

2.0 The Qoda Protocol

Qoda enables collateralized borrowing and lending of crypto assets at fixed interest rates for fixed periods, where terms are set by the users themselves. It is implemented similar to a futures exchange, except the quoted figure is the APR of the loan instead of the token price.

2.1 Gasless Quotes

The *Quote* is Qoda's analogue to limit orders in an orderbook. Users publish *Quotes* into the platform, indicating whether they are a borrower or lender, what rate they wish to deal at, for what size, and until which maturity date. Importantly, a *Quote* can theoretically be generated completely off-chain via the browser or the *Quoter*'s local machine. The reasoning for this mechanism is to provide gas savings for users – only the bare minimum of trade executions ever need to hit the blockchain.

The basic structure of a Quote looks like this:

```
Quote {
  address marketAddress, // Address of the FixedRateLoanMarket contract
  address quoter, // Public address of the Quoter
  uint8 quoteType, // 0 for PV+APR, 1 for FV+APR
  uint8 side, // 0 if Quoter is borrowing, 1 if Quoter is lending
  uint64 quoteExpiryTime, // Timestamp after which Quote is no longer valid
  uint64 APR, // Equivalent yearly simple interest on PV
  uint256 cashflow, // PV if quoteType=0, FV if quoteType=1
  uint256 nonce, // To guarantee uniqueness of Quote, prevent signature replay attacks
  bytes signature // Used for verifying Quote
}
```

The key field is the s*ignature*. The *Quoter* hashes all the other fields of the *Quote* and signs that hash with their private key, generating the *signature*. The *signature* trustlessly proves that the *Quoter* is in fact willing to enter the loan at the specified terms. A *Responder* to a *Quote* needs to enter all the fields of the *Quote* together with the accompanying *signature* in order to transact. If the fields do not match the signature, the transaction will revert.

Given two of following fields, one can always calculate the third: *Present Value* (PV), *Future Value* (FV), and *Annual Percentage Rate* (APR). In a *Quote*, the user is required to input a *cashflow* value, which can either represent the PV or the FV. This is determined by the *quoteType* field. This way, the *Quoter* is always able to input his desired size conveniently either in terms of the present value or the future value. APR is a required field so that the equivalent rate always stays constant over time.

2.2 Maturities

Maturities are UNIX timestamps that represent the settlement date of loans. Maturity dates must be explicitly enabled on the platform before it becomes a valid date. To start, these will be quarterly dates similar to futures exchange offerings, i.e. 31st March, 30th June, 30th September, 31st December 12:00am GMT, with the flexibility to add more tradeable dates if desired.

2.3 Assets

Each ERC20 token supported by Qoda is mapped one-to-one to an *Asset*. Its main purpose is to define the scope for deposit of collateral. The *Asset* struct has the following fields:

```
Asset {
  bool isEnabled, // Must be true for the Asset to be enabled
  address oracleFeed // Address of the associated Chainlink oracle price feed
  uint collateralFactor // Value between 0 - 1, used to discount the value of the
                        //  collateral provided by a borrower
  uint marketFactor // Value between 0 -1, used to apply a premium on the value of
                    // assets borrowed
  uint[] maturities // A list of all the enabled maturity dates for this Asset
}
```

An *Asset* must be enabled before users may deposit collateral denominated in the ERC20 token. For more information on this, see *Collateral Management* (section 2.5).

2.4 Markets

Each *FixedRateMarket* is a separate smart contract deployment characterized by two parameters: *tokenAddress*, the address of the ERC20 token which the loan will be denominated in, and *maturity*, the UNIX timestamp representing the settlement date of the loan.

Each *FixedRateMarket* is itself an instance of ERC20 and has its own associated *qToken*. For example:

> *A user lends 100 GLMR at 10% fixed rate, expiring on 31 March, 2022. When the transaction is executed, the system will mint 110 qGLMRMAR22 tokens to the user. Upon the expiry of the contract on 31st March 1:00am, the 110 qGLMRMAR22 tokens will be redeemable for the underlying 110 GLMR tokens at a 1:1 rate.*

The advantage of *qTokens* is that it allows for greater capital allocation efficiency. While the user can simply hold onto them, the *qTokens* themselves hold value and can potentially be traded in secondary markets, staked in yield farms, used as collateral itself in other protocols, etc.

Note that the *qToken* is not 100% freely transferrable. Every *FixedRateMarket* also maintains a mapping of *accountBorrows* in storage for how much each user has borrowed in total. Users can only transfer *qTokens* that are in excess of their current borrows. For example, an account that only lends and never borrows can freely transfer their tokens, but an account that owns 110 qGLMRMAR22 tokens but has also borrowed 100 GLMR expiring March 2022 may only transfer out 10 qGLMRMAR22. This is to protect the protocol from users gaming the collateral management system by borrowing off of the *qToken* and then immediately transferring out the *qToken* to another address, leaving the borrowing account uncollateralized.

2.5 Collateral Management

The net borrows of any account must always be overcollateralized at all times to ensure nondefault. Any account in danger of undercollateralization is subject to liquidations as described in section 2.7. The overall collateral health of an account, called *liquidityRatio* depends on two components:

1) collateralValue Before an account is allowed to take on a loan, they must fund it with collateral, which can be denominated in any token that is an enabled *Asset* (section 2.3). Its value is expressed in USD terms using Chainlink price feeds, calculated as:

$$\sum_{Asset} amount_{Asset} \times exchRate_{Asset/USD} \times collateralFactor_{Asset}$$

where *collateralFactor* is a parametrized value from 0.0 to 1.0 based on the *Asset.* The value will be higher for safer *Assets*, and lower for riskier *Assets*. Note this means that the *collateralValue* of an account will always be lower than the actual market value of the underlying tokens. Hence, the *collateralFactor* parameter ensures that account borrows will always be overcollateralized.

2) borrowValue The sum of all the borrows of an account across all *Markets*. This is calculated as:

$$\sum_{Market} max[borrow_{Market} - qToken_{Market}] \times exchRate_{Market/USD} \div marketFactor_{Market}$$

Note that the *borrow* amount refers to the full principal plus interest amount (i.e. Future Value, FV), not just the principal amount upon inception of the loan (ie.e. Present Value, PV). The amount of *qTokens* for a particular *Market* also act as a credit to the user for that *Market*. The feature of

netting off *borrow* with *qTokens* makes borrows and lends fungible for each *Market,* which give users the flexibility to trade in and out of positions.

Similar to the *collateralFactor,* the *marketFactor* is a parametrized value from 0.0 to 1.0 based on the *Asset*, resulting in the *borrowValue* of an account to always be valued at a premium compared to the actual market value of the underlying token. This again ensures that account borrows will always be overcollateralized.

The account is considered properly collateralized as long as the ratio of c*ollateralValue / borrowValue* remains above 1. If an account falls below this, it will be subject to liquidation.

2.6 Settlement of Loans

Borrowers may repay their borrows for any *Market* at any time before its maturity date. They may either repay with the underlying token, or with the corresponding *qToken*. The repayments are held in escrow until the maturity date. In order to encourage timely repayment of borrows, any account that has not paid after the maturity date block has been finalized is subject to the same liquidation procedures described in section 2.7, even if the account is still overcollateralized.

At 12:00am on the maturity date, all trading for that *Market* will cease. There will be a grace period of 1 hour to allow enough time for borrowers to make their repayments, or for *liquidators* to pay on behalf of any non-payers, so that the smart contract funds are sufficient for lenders for withdrawal. At 1:00am, the lenders may burn their *qTokens* to redeem the underlying from the smart contract at a 1:1 exchange rate.

2.7 Liquidations

When an account's *liquidityRatio* falls below 1 or if it is late in its repayments, it is subject to liquidations. In this scenario, a third-party, called the *liquidator*, can repay the full loan amount on behalf of the borrower. The equivalent USD value of the borrower's collateral plus a parameterized percentage of the value, called the *liquidationPenalty,* will be released to the liquidator as an economic incentive for maintaining the collateral health of the system as a whole. The onus will be on the *liquidator* to swap the received collateral back to the currency of the loan if they wish to crystallize their profits. To bootstrap the protocol, Qoda will also be running its own liquidation bot, but it is expected to quickly be outpaced by third party liquidation bots with more sophisticated strategies when the economic motivation becomes apparent.

3.0 Future Roadmap

The long-term goal for Qoda is to establish itself as a powerhouse for fixed income – *a decentralized investment bank that connects those who need capital to those who have capital.* To that end, there are several exciting future products in the pipeline on Qoda's long-term roapmap. These include pegged floating rate loans (e.g., Compound/Aave mid rates, perpetual futures funding rates, etc.), over-the-counter derivative products such as fixed-for-floating interest rate swaps, decentralized corporate bond origination and secondary trading for crypto businesses, and credit default swaps. More details to follow on this.