1.0 The Current Landscape of DeFi Interest Rates

The most popular collateralized lending/borrowing protocols today generally offer floating interest rates via pools which users can supply or borrow from. While these protocols create a lot of value for users looking to deploy their capital, there is increasingly more demand both from lenders and borrowers for more predictable, fixed interest rates. Furthermore, with fixed rate deals recorded on-chain across a variety of matrutiy dates, one can start to construct a new DeFi primitive, the crypto *Yield 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.

Shoehorning support for fixed interest rates using AMM-style liquidity pools has proven to be a difficult task so far. One reason is that it can be difficult to correctly model what price impact / slippage should look like for rates. Another is that with the sheer number of markets for a single token (e.g. Quarterly, monthly, weekly maturity dates) – it will quickly get very capital intensive for liquidity providers who need to put up massive sums of capital to keep such a large number of markets liquid. In tradfi, orderbooks or auction-based marketplaces avoided these issues because 1) the limit order system properly reflects the non-linear, time-dependent nature of real supply and demand and 2) capital doesn't need to put up front when placing orders, only when trades are executed. Orderbooks are infeasible for fast-moving cryptocurrency markets due to transaction fees and minimum block times, but the marketplace model has shown to be effective for slower, more illiquid products such as NFTs (e.g. Opensea). Similarly, a marketplace model can be effective for a slower moving product such as fixed income.

Furthermore, while there are protocols that offer variable yield staking for tokens, and numerous protocols that offer loans denominated in stablecoin, there are perhaps no protocols today that offer fixed interest rate borrowing/lending of altcoins. This is important because the ability to borrow altcoins enables users to short the token (i.e. borrow and sell the token today for stablecoin, and buy back the token in the future at a lower rate). This type of market 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.

With these three issues in mind, we introduce the Qoda protocol, a collateralized lending/borrowing protocol differentiating itself with three features: fixed interest rates over fixed maturity dates, implementing a marketplace model instead of AMM, and a focus on altcoin-denominated loans. The long-term mission of the protocol is to grow the crypto interest rates market to become as popular and mature as the crypto currency markets are today by *building a complete fixed income solution for 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 FixedRateMarket 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 a user may not have an outstanding debt and hold *qTokens* at the same time. Every *FixedRateMarket* also maintains a mapping of *accountBorrows* in storage for how much each user has borrowed in total. If a user with an outstanding loan is given *qTokens*, the balance of the loan will be deducted first – the user will only receive *qTokens* that are in excess of their current borrows. Similarly, if a user with outstanding borrows in a *Market* subsequently lends into the *Market*, the balance of the loan will be deducted first and they will only be minted new *qTokens* in excess of their current borrows.

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) virtualCollateralValue 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* and is used to discount the value of the collateral. The value will be higher for safer *Assets*, and lower for riskier *Assets*. Note this means that the *virtualCollateralValue* of an account will always be lower than the actual market value (or *realCollateralValue)* of the underlying tokens. Hence, the *collateralFactor* parameter ensures that account borrows will always be overcollateralized.

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

$$\sum_{Market} borrowAmount_{Market} \times exchRate_{Market/USD} \div marketFactor_{Market}$$

Note that the *borrowAmount* refers to the full principal plus interest amount (i.e. Future Value, FV), not just the principal amount upon inception of the loan (i.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*, which is used to give a premium on value of the user's loans. The value will be higher for safer *Markets*, and lower for more volatile *Markets.* Therefore, the *virtualCollateralValue* of an account will always be valued at a premium compared to the actual market value (or *realBorrowValue*) of its underlying token. This again ensures that account borrows will always be overcollateralized.

The account is considered properly collateralized as long as the ratio of *virtualCollateralValue / virtualBorrowValue* 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, along with its associated liquidation penalties, as 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 *liquidationIncentive* 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.

2.8 Protocol Fees

The protocol takes an annualized fee of 20 basis points (subject to governance changes) on all executed loans. The fee is applied to both sides – both to the lender and the borrower – and is pro-rated linearly as a function of the maturity of the loan.

The most natural way to take fees is in the currency of the denomination of the loan. However, rather than requiring staked tokenholders (see section 3.0) to claim the protocol fees in various different currencies, the protocol will automatically convert all accrued fees on any given Market (once it has reached a meaningful amount) to a common base currency, e.g. GLMR, and distribute rewards to stakers in this common currency.

3.0 Tokenomics & Token Utility

QODA is the native token of the protocol. There is a fixed, fully-diluted, non-inflationary supply of 1,000,000,000 QODA tokens. The native QODA token confers no additional utility or rights, but can be staked for veQODA, inspired by the tokenomics design of Platypus Finance.

By default, each 1 QODA token that is staked will generate 0.27 veQODA per day, until it reaches a cap of 100x the initial staked amount, i.e. a max of 100 veQODA tokens for every staked QODA. It will take roughly 1 year for the staked token to reach the veQODA cap. veQODA tokens will be neither transferrable nor tradable, i.e. locked in the staker's wallet, and users may unstake their QODA tokens at any point. However, any accumulated veQODA tokens immediately drops to 0 as soon as the user unstakes any amount. This incentivizes users to remain long-term aligned with the goals of the protocol, while at the same time providing the flexibility for liquid staking and unstaking without any minimum staking period.

The veQODA tokens confer three primary benefits:

1) A share of all protocol fees as a percentage of the full current supply of veQODA.
2) Token emissions of QODA, distributed as a percentage of the full current supply of veQODA.
3) Governance DAO rights

All veQODA parameters are subject to modification via governance, and more features may be added in the future.

3.1 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.