Alpha Lending

Decentralized pool-based lending protocol driving cross-chain DeFi and cross-chain liquidity

Whitepaper Version 1.0
September 2020

Abstract

Alpha Lending is built by the Alpha Finance Lab team. Alpha Finance Lab is a DeFi lab, focusing on building an ecosystem of innovative Alpha products that will interoperate. Innovation is the core of Alpha, and we are launching products quickly with the goal of generating optimal alpha to users. Depending on the nature and go-to-market strategy of each product, some products will be deployed on Ethereum and some on Binance Smart Chain. Alpha is building a cross-chain DeFi platform across Ethereum and Binance Smart Chain.

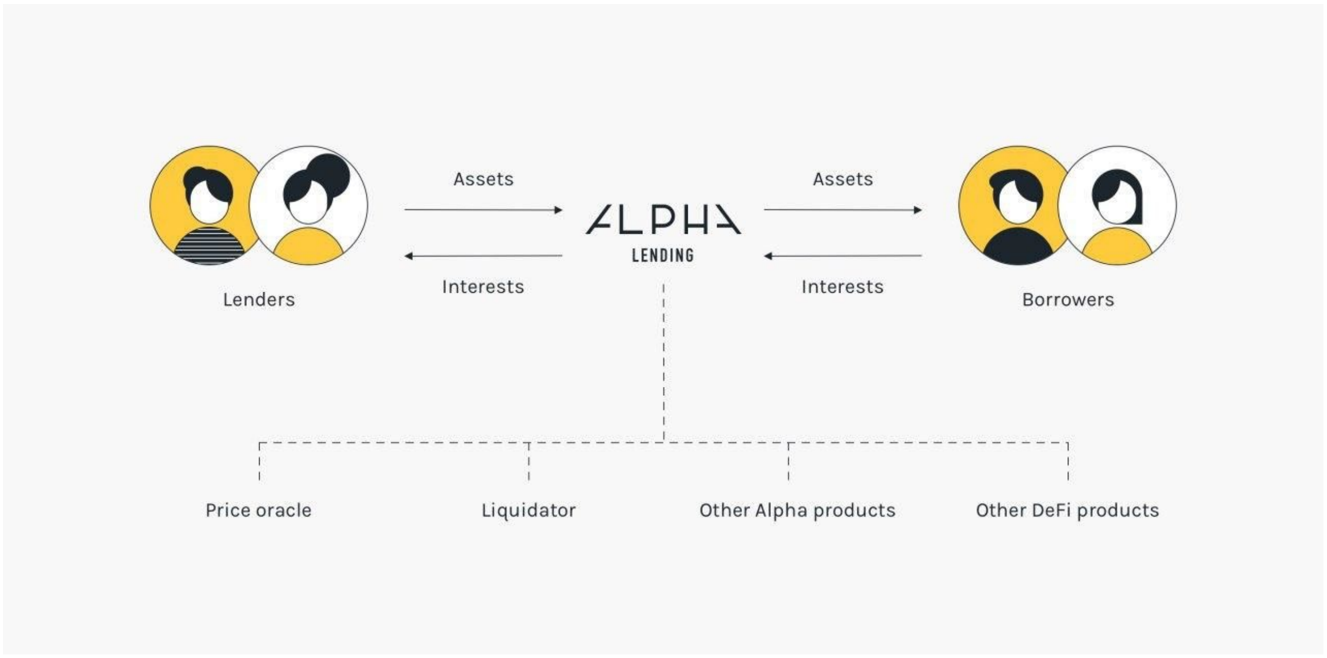
This whitepaper explains terminologies, theories, and protocol architecture behind Alpha Lending.

Contents

1. Alpha Lending Overview

Alpha Lending is a decentralized pool-based lending protocol built on Binance Smart Chain. Users can earn interests on their digital assets (assets) by supplying supported assets into the protocol. We call these users lenders. Assets deposited by lenders will be transferred into a smart contract that aggregates total liquidity of each asset into a pooled fund, which is available for borrowers to borrow. Loans are not matched individually between lenders and borrowers, but are taken from the pooled fund. Interests lenders earn come from interests that borrowers pay, distributed proportionately to the liquidity they provide.

Once lenders supply assets into the protocol, these assets act as collateral, enabling lenders to also borrow any asset up to a certain limit. This means that any borrower has to first supply assets into the protocol as collateral before taking out any loan.



2. Terminology

Total Liquidity: Total amount of liquidity in a each asset pool, calculated as *Total Available Liquidity + Total Borrows*

Total Available Liquidity: Total liquidity available of an asset for borrowers to borrow or lenders to withdraw

Total Borrows: Sum of total borrowed amount and accumulated borrow interest of an asset

alToken: A token that represents the user's lending position which equates to the share of deposited amount contributed to *Total Liquidity* of that asset

Total alToken: Total number of shares of *Total Liquidity* of that asset

Asset Maximum Loan-to-value (LTV): Maximum percentage of deposited asset that can be borrowed out

Borrow Limit: Total value that a user can borrow, determined based on the total collateral deposited and the *Asset Maximum LTV* of each deposited asset

Account Health: *Account Health* is healthy if total value of borrowed assets is below the Borrow Limit

Borrow Shares: Share of the user's borrowed amount to *Total Borrows* of that asset

Total Borrow Shares: Total number of shares of *Total Borrows* for an asset.

Withdraw Shares: Number of shares calculated based on the inputted withdrawal amount. Number of *alTokens* is to be reduced by this *Withdraw Shares* when a user withdraws the deposited asset

Repay Shares: Number of shares calculated based on the inputted repay amount. Number of *Borrow Shares* is to be reduced by this *Repay Shares* when a user repays the borrowed asset

Utilization Rate: *Total Borrows / Total Liquidity* which reflects the demand to borrow an asset.

Borrow Interest Rate$_1$: The interest rate the borrowers pay for borrowing an asset when *Utilization Rate* is below the *Optimal Utilization Rate*. Calculated as *Base Borrow Rate + (Utilization Rate * Slope$_1$)*

Borrow Interest Rate$_2$: The interest rate the borrowers pay for borrowing an asset when *Utilization Rate* is above the *Optimal Utilization Rate*. Calculated as *Borrow Interest Rate = Slope$_1$ + [(Utilization Rate - Optimal Utilization Rate)/(100% - Optimal Utilization Rate) * Slope$_2$]*

Optimal Utilization Rate: Specific *Utilization Rate* that marks the beginning of a sharp rise in *Borrow Interest Rate*

Deposit Interest Rate: The interest rate the lenders receive from depositing their assets. Calculated as *Borrow Interest Rate * Utilization Rate*

Pool Reserve: A portion of *Borrow Interest Rate* allocated as an insurance for the pool

Close Factor: The portion of the borrowed asset that can be repaid by a liquidator in one transaction

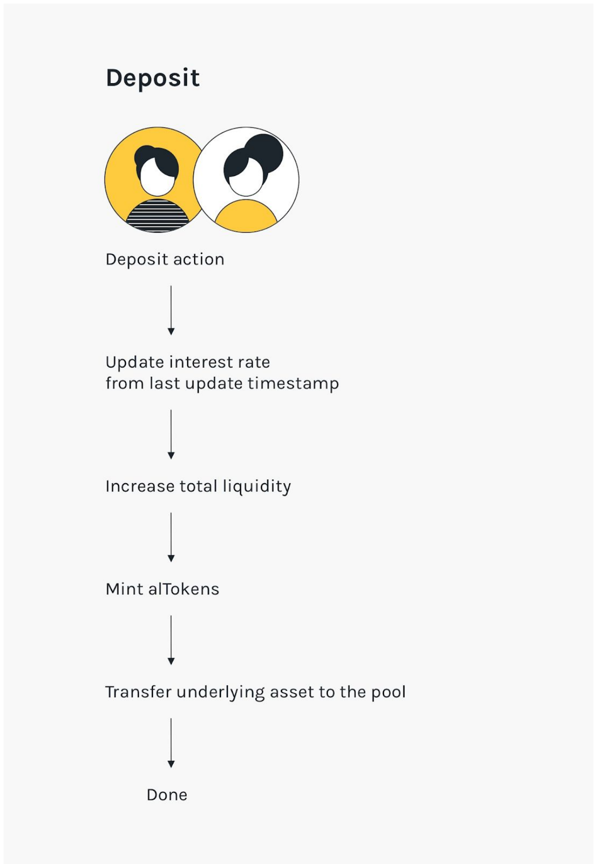
Liquidate Shares: Number of shares calculated based on the inputted liquidate amount. *Borrow Shares* is to be reduced by *Liquidate Shares* when a liquidator repays the borrowed asset

Collateral Amount: The value of a user's collateral asset equivalent to the liquidate value at a discounted price

Liquidation Bonus: The difference between *Collateral Amount* that the liquidator receives and the liquidate amount the liquidator pays

3. Key Concepts

3.1 Deposit



Here is an example of the diagram above. Alice first deposits one of the supported assets, such as BNB, into the protocol. Once deposited, the deposited BNB is added into a pooled fund, which is referred to as the ***Total Liquidity***. This *Total Liquidity* is calculated as follows:

$$\textit{Total Liquidity} = \textit{Total Available Liquidity} + \textit{Total Borrows}$$

Total Available Liquidity is the liquidity available of that asset for borrowers to borrow or lenders to withdraw. The ***Total Borrows*** of an asset is the sum of the total borrowed amount and the accumulated borrow interests from all borrowers. *Total Borrows* is calculated as follows:

$$\textit{Total Borrows} = \textit{Borrow Amount} + \textit{Cumulative Borrow Interest}$$

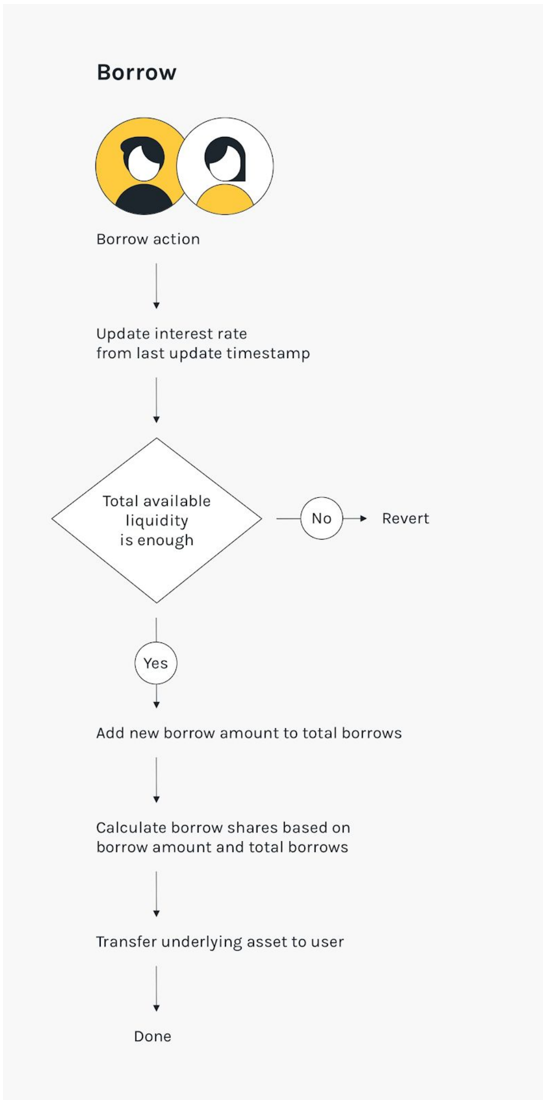
This means that *Total Liquidity* will continue to grow as *Cumulative Borrow Interest* grows over-time. More details on interest rate can be found in section 3.5. Interest Rate Dynamics.

Alice will receive ***alTokens***, such as alBNB, that represent her share of BNB deposited to *Total Liquidity* of BNB. *alToken* is a tokenized representation of the user's lending position, and is an interest-bearing BEP20 token, which means an *alToken* can claim more of the underlying asset over-time as *Total Liquidity* grows from interest collected from the borrowers. Number of *alTokens* each user receives is calculated as follows:

*Number of alTokens = Deposit Amount * Total alToken / Total Liquidity*

Total alToken is set based on the first user who deposits this asset. For example, if Bob is the first user who deposits 1,000 BNB, then *Total alBNB* starts with 1,000 alBNB. If Alice then adds 100 BNB when the *Total Liquidity* is 1,000 BNB, then Alice gets 100 alBNB (100 * 1,000 / 1,000).

3.2 Borrow


Borrow
Borrow action
Update interest rate from last update timestamp
Total available liquidity is enough
No
Revert
Yes
Add new borrow amount to total borrows
Calculate borrow shares based on borrow amount and total borrows
Transfer underlying asset to user
Done

Before borrowing

Before a user can borrow, they have to first deposit some of their assets that can be used as collateral on the protocol**.** Upon depositing such assets, the user receives *alTokens,* representing the user's shares in the assets' pools. . Note that some assets are not accepted as collateral in order to protect the protocol's security. Even though these *alTokens* are used as collateral, the user will still earn deposit interest on them since other users are borrowing the underlying asset from the asset pool and paying borrow interest to the pool, or *Total Liquidity*.

For example, Alice can deposit BUSD, one of the available collateral assets, into the protocol and receives a balance of alBUSD that represents her share in the total BUSD pool . Alice can then use this alBUSD as collateral*,* enabling her to borrow other assets such as ETH. In this case, Alice is earning deposit interest on BUSD and paying borrow interest on ETH. More details on interest rates can be found in section 3.5. Interest Rate Dynamics.

Each asset that can be used as collateral has an assigned ***Asset Maximum Loan-to-value (LTV)***. For instance, if Alice deposits $100 worth of BUSD, which has an *Asset Maximum LTV* of 75%, then Alice can borrow any asset with the ***Borrow Limit*** of $75. *Borrow Limit* is calculated based on the total value of deposited assets that can be used as collateral and *Asset Maximum LTV* of each deposited asset. Specifically, the *Borrow Limit* for an asset can be calculated as follows:

$$\textit{Borrow Limit} = (\textit{Deposit Value in USD of Asset}_1 * \textit{Asset Maximum LTV}_1 + \textit{Deposit Value in USD for Asset}_2 * \textit{Asset Maximum LTV}_2 + \ldots)$$

A user can only borrow if the ***Account Health*** remains healthy after taking into account the new borrow amount. *Account Health can be* calculated as follows:

$$\textit{Account Health} = \textit{Healthy (borrow value} \leq \textit{Borrow Limit)}$$
$$\textit{Account Health} = \textit{Unhealthy (borrow value} > \textit{Borrow Limit)}$$

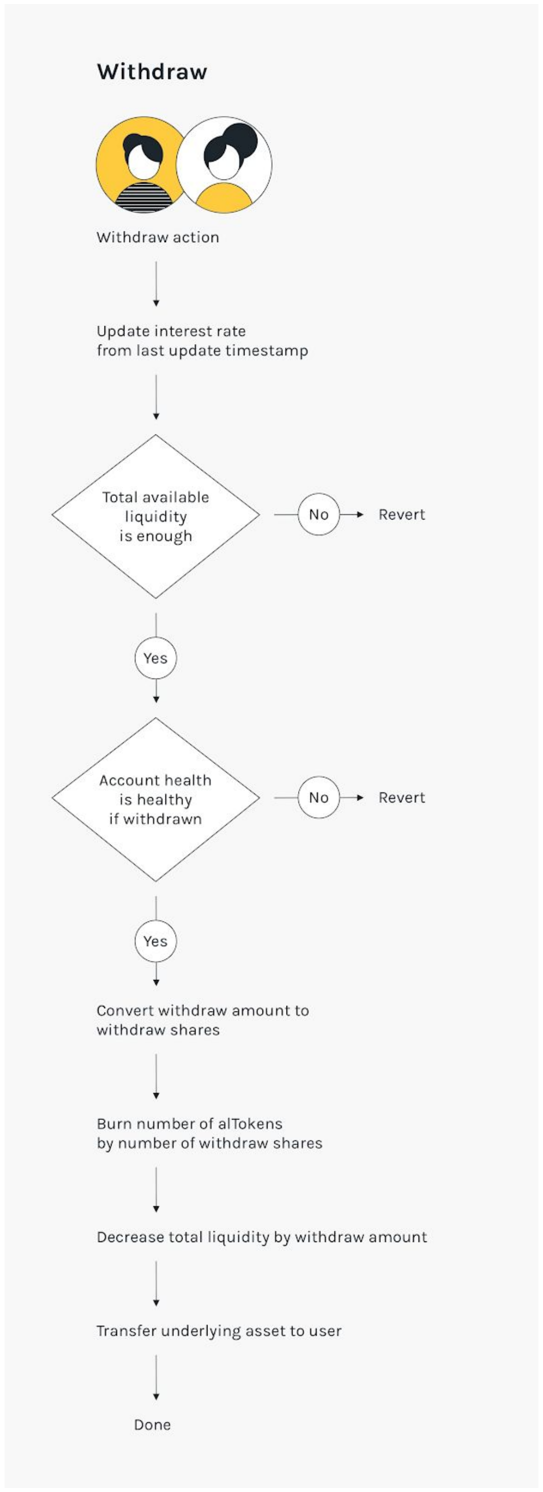
Borrowing process

When a user borrows and receives the borrowed amount, the protocol calculates how many ***Borrow Shares*** the borrowed amount equals to. *Borrow Shares* represent the share of the user's borrowed amount to the *Total Borrows* of that asset. *Borrow Shares* is calculated as follows:

$$\textit{Borrow Shares} = (\textit{Borrow Amount} * \textit{Total Borrow Shares}) / \textit{Total Borrows}$$

The number of ***Total Borrow Shares*** is set based on the first user who borrows this asset. For example, if Bob is the first user who borrows 1,000 BNB, then the number of *Total Borrow Shares* starts with 1,000. If Alice then borrows 100 BNB when the *Total Borrows* is 1,000 BNB, then Alice gets 100 *Borrow Shares* (100 * 1,000 / 1,000).

3.3 Withdraw


Withdraw
Withdraw action
Update interest rate from last update timestamp
Total available liquidity is enough
No
Revert
Yes
Account health is healthy if withdrawn
No
Revert
Yes
Convert withdraw amount to withdraw shares
Burn number of alTokens by number of withdraw shares
Decrease total liquidity by withdraw amount
Transfer underlying asset to user
Done

Before withdrawing

A user can withdraw the amount only if there is enough *Total Available Liquidity* to do so and if the *Account Health* remains healthy after the transaction.

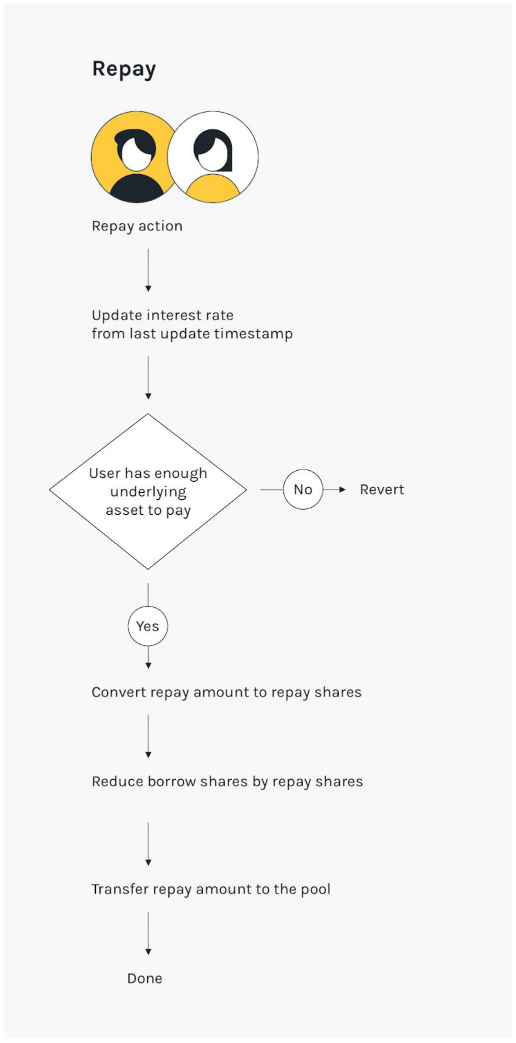
Withdrawing process

To withdraw a part or all of the deposited amount, the protocol calculates ***Withdraw Shares*** from the withdraw amount inputted, burns *alTokens* equal to the number of *Withdraw Shares,* and transfers the withdraw amount to the user. *Withdraw Shares* is calculated as follows:

$$\textit{Withdraw Shares} = \textit{Withdraw Amount} * \textit{Total alToken} / \textit{Total Liquidity}$$

Because *Total Liquidity* increases over-time from accruing borrow interests, the same withdrawal amount will equal smaller *Withdraw Shares* over-time, and thus burns fewer *alTokens* to claim the same withdraw amount. If the user withdraws all of the deposited amount, the user will receive a withdraw amount that is more than the originally deposited amount from accruing deposit interests. More details on interest rates can be found in section 3.5. Interest Rate Dynamics.

3.4 Repay


Repay
Repay action
Update interest rate from last update timestamp
User has enough underlying asset to pay
No
Revert
Yes
Convert repay amount to repay shares
Reduce borrow shares by repay shares
Transfer repay amount to the pool
Done

To repay a part or all of the borrowed amount, the protocol calculates ***Repay Shares*** from the repay amount inputted, transfers the repay amount to the pool, and reduces *Borrow Shares* by *Repay Shares*. *Repay Shares* is calculated as follows:

$$\text{Repay Shares} = \text{Repay Amount} * \text{Total Borrow Shares} / \text{Total Borrows}$$

Because *Total Borrows* increases over-time from accruing borrow interests, the same repay amount will equal smaller *Repay Shares* over-time, reducing *Borrow Shares* by a smaller *Repay Shares*. If the user repays all of the borrowed amount, the user will pay more than the original amount because of accrued borrow interests. More details on interest rate can be found in section 3.5. Interest Rate Dynamics.

3.5 Interest Rate Dynamics

Interest rates for borrowers and lenders are determined by ***Utilization Rate***. *Utilization Rate* is calculated as follows:

$$\text{Utilization Rate} = \text{Total Borrows} / \text{Total Liquidity}$$

Because the *Utilization Rate* reflects the demand to borrow an asset, a higher *Utilization Rate* corresponds to a higher cost of borrowing or borrow interest rate. Each asset has its own base borrow rate and ***Optimal Utilization Rate***, or the specific *Utilization Rate* that marks the beginning of a sharp rise in *Borrow Interest Rate* to protect the liquidity in the pool. Therefore, ***Borrow Interest Rate***$_1$ and ***Borrow Interest Rate***$_2$ when *Utilization Rate* is below and above *Optimal Utilization Rate* can be calculated as:

Borrow Interest Rate$_1$ when Utilization Rate < Optimal Utilization Rate:

$$\text{Borrow Interest Rate} = \text{Base Borrow Rate} + (\text{Utilization Rate} * \text{Slope}_1)$$

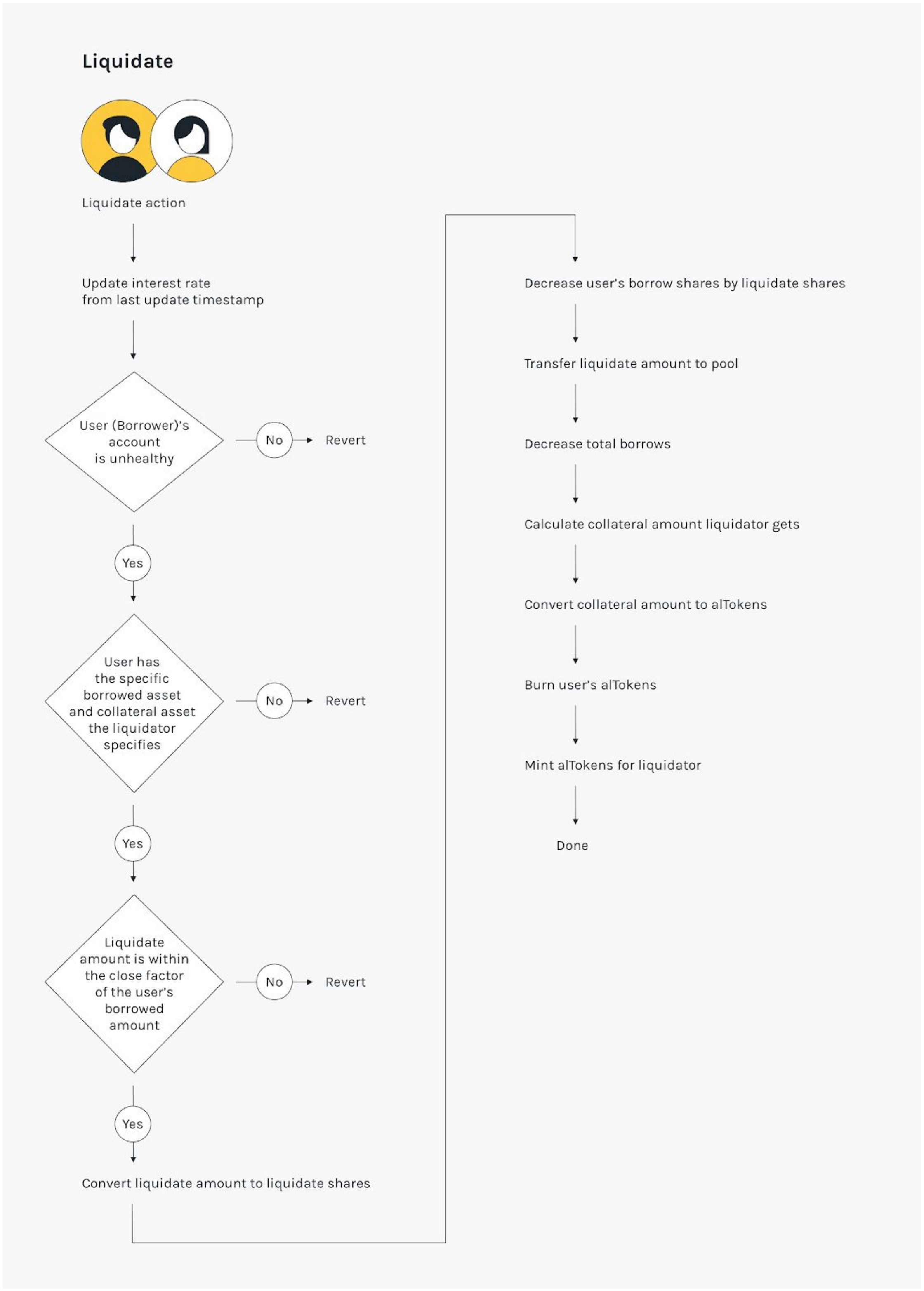
Borrow Interest Rate$_2$ when Utilization Rate > Optimal Utilization Rate:

$$\text{Borrow Interest Rate} = \text{Slope}_1 + [(\text{Utilization Rate} - \text{Optimal Utilization Rate})/(100\% - \text{Optimal Utilization Rate}) * \text{Slope}_2]$$

5-10% of the *Borrow Interest Rate* will be allocated for ***Pool Reserve*** as an insurance for the pool. Since the accumulated borrow interests are added to *Total Liquidity* and the *alTokens* that lenders receive can claim the share of *Total Liquidity,* the higher borrow interest rate corresponds to the higher ***Deposit Interest Rate***, which can be calculated as:

$$\text{Deposit Interest Rate} = \text{Borrow Interest Rate} * \text{Utilization Rate}$$

4. Risk And Liquidation


Liquidate
Liquidate action
Update interest rate from last update timestamp
User (Borrower)'s account is unhealthy
No
Revert
Yes
User has the specific borrowed asset and collateral asset the liquidator specifies
No
Revert
Yes
Liquidate amount is within the close factor of the user's borrowed amount
No
Revert
Yes
Convert liquidate amount to liquidate shares
Decrease user's borrow shares by liquidate shares
Transfer liquidate amount to pool
Decrease total borrows
Calculate collateral amount liquidator gets
Convert collateral amount to alTokens
Burn user's alTokens
Mint alTokens for liquidator
Done

Causes of liquidation

A borrower bears the risk of having an unhealthy *Account Health* when the user's total value of borrowed assets exceeds the *Borrow Limit*. Volatility in collateral assets and borrowed assets can cause *Account Health* to be unhealthy.

For instance, Alice deposits $100 worth of BUSD, which for instance has an *Asset Maximum LTV* of 75%, enabling Alice to borrow any asset up to the *Borrow Limit* of $75, and borrows $75 of ETH (e.g. 0.18 ETH) when ETH price is $400. If ETH price increases afterwards, Alice's total value of borrowed assets increases since 0.18 ETH now equals to more than $75, resulting in her total value of borrowed assets being higher than her *Borrow Limit*. On the other hand, *Account Health* can become unhealthy when the price of the collateral asset, or in this case BUSD, decreases such that *Borrow Limit* becomes less than the total value of borrowed assets.

Due to the volatile nature of asset prices, users are recommended to borrow at a value less than the *Borrow Limit* to avoid liquidation.

Liquidation process

When *Account Health* becomes unhealthy, any external actor called a liquidator can repay up to the ***Close Factor*** of the user's borrowed amount. This *Close Factor* is the portion of the borrowed asset that can be repaid by a liquidator in one transaction. The liquidation process may continue until the user's *Account Health* becomes healthy, or when the total value of borrowed assets is below *Borrow Limit*. The *Close Factor* ensures that the user's account will not be fully liquidated if not necessary.

When a liquidator repays the user's borrowed amount, liquidator inputs ***Liquidate Shares*** and the protocol reduces the user's *Borrow Shares* by *Liquidate Shares*.

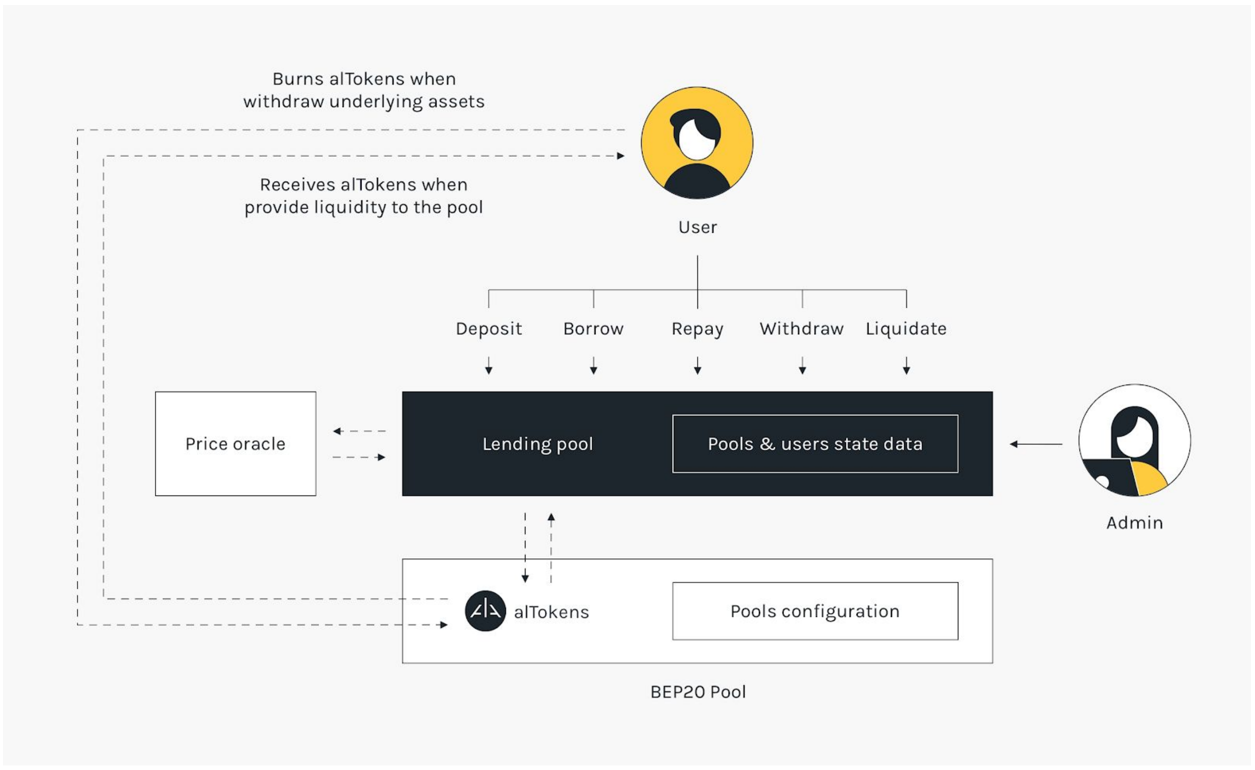
To reward the liquidator for liquidating an unhealthy account, the liquidator can purchase ***Collateral Amount***, or the value of the user's collateral asset equivalent to the liquidate value at a discounted price. The difference between *Collateral Amount* that the liquidator receives and the liquidate amount the liquidator pays is captured by ***Liquidation Bonus***. *Collateral Amount* is calculated as follows:

*Collateral Amount = (Liquidate Amount * Price of Liquidated Asset / Price of Collateral Asset) * (Liquidation Bonus)*

Liquidate Amount is calculated as follows:

*Liquidate Amount = (Liquidate Share * Total Borrows) / Total Borrow Shares*

5. Protocol Architecture



<u>5.1 Lending Pool</u>

Lending Pool is the core contract of the protocol . This contract manages all states and handles user interactions with the lending pool. Interactions include deposit, borrow, repay, withdraw and liquidate.

<u>5.2 Lending Pool Configurator</u>

Lending Pool Configurator provides configuration functions for Lending Pool and implements the configuration of BEP20 token pools. Configurations include pool configuration, and interest rates calculation.

<u>5.3 alToken</u>

alToken contract manages the minting and burning of alTokens. alToken represents a user's lending position or the share of deposited amount to *Total Liquidity* of that asset.

<u>5.4 Price Oracle</u>

The protocol uses Band Protocol's BandChain oracle. Price Oracle contract is responsible for querying the latest price of an asset from BandChain.

6. ALPHA Governance Token

ALPHA will be critical to not only bootstrap liquidity, but also act as the main utility token across a portfolio of Alpha products, including staking utilities, reward systems, as well as governance mechanisms. All Alpha products will tie back to ALPHA tokens, rewarding users for using our products by taking the revenues from the products to buyback or burn ALPHA tokens.

As Alpha Finance Lab continues to innovate in the DeFi space and build more Alpha products, ALPHA token holders will be able to propose and decide upon key changes to all Alpha products and how they interoperate. ALPHA tokens will continue to be the primary tool to align incentives of community builders and supporters, creating a strong community of DeFi enthusiasts who will collectively help propel Alpha ecosystem forward.