

# Clearpool Prime

## Abstract

In this paper we introduce a decentralized peer-to-peer financial instrument which functions as the facilitator and record keeper for agreements and transactions made between a whitelisted group of participants. Clearpool Prime enables vetted and verified institutions to borrow USDC (or other ERC-20 assets) from one or more of a pool of lenders, selected by the borrower, for a fixed time period and interest rate.

## Participants

- Due Diligence Provider (“DD Provider”)
- Governor
- Borrower
- Lender

## Smart-contract whitelist

The smart contract whitelist is required to ensure the integrity of the protocol, allowing only Prime participants to use the instrument. After successfully passing the KYC/KYB procedure with DD Provider, the participant(s) will apply for a smart contract whitelist. The Protocol Governor approves or rejects whitelist applications.

## Implementation & architecture

Clearpool Prime is an EVM compatible protocol that matches the peer-to-peer orders of its whitelisted participants. The participants, suppliers and borrowers of an asset, interact directly with the protocol’s smart contracts, earning (and paying) fixed interest rates without the requirement for collateral, custody or intermediary. All activities within the protocol are transparent and publicly inspectable.

## Borrowing

Borrowing with Prime does not require the provision of collateral. A borrower places an order with specific terms within the core smart-contract by creating a Pool. When the Pool is created, the borrower selects from all Prime members the specific lenders whom he wants to whitelist and invites them to fund the Pool. Borrowers can also opt-in to make the pool public, therefore, all Prime whitelisted members will have explicit access to the pool. Clearpool Prime does not

hold custody of the loaned assets, instead, all funds are transferred automatically to the Borrower's wallet and a record to match it is created within the smart-contract.

## Pool

Pools are the settlement venue for borrowing in the Clearpool Prime protocol. Borrowers create pools to receive funding from lenders according to agreed upon economic terms. Two types of pools are currently supported across the protocol. These include Bullet and Monthly Repayment pools. Borrowers are allowed to have multiple Pools opened simultaneously.

### Bullet Repayment Pools

Bullet repayment Pools must be repaid completely prior to maturity date. The borrower can repay the Pool at any time either to all lenders collectively or to any specific lender.

### Monthly Repayment Pools

Monthly repayment Pools require interest repayment every 30 days. The first interest repayment is due exactly 30 days from first lend() transaction into the pool, at which date all liquidity providers will be paid interest for the amount of time their liquidity has been in the pool.

## Deployment

Pools are deployed using the Pool Factory smart contract. Only smart-contract whitelisted users and the Protocol Governor are capable of interacting with the Factory smart-contract. When deploying a new Pool, the borrower specifies the following terms:

- a. **Repayment option** - bullet or monthly repayment
- b. **Asset** - type of pull asset (USDC/USDT/DAI)
- c. **Size** - the maximum amount of assets to be borrowed
- d. **APR** - borrowing APR
- e. **Tenor** - the time range of the maturity of the loan (Year = 360 days, Month = 30)
- f. **Rate** - the percentage of interest offered by borrower
- g. **Deposit Window** - the time range within which the Borrower can accept liquidity after the first lend transaction

After deployment, the Pool is *open* but *inactive* until the first lender provides liquidity. Once liquidity is provided, the **lend()** function is called within Pool smart-contract and the **Maturity date** as well as Deposit Window date are defined.

## Data Structure

data	type	description
asset	enum	USDC / DAI / USDT
maxSize	int	Loan asset cap
currentSize	int	Used to track how much of the Pool has already been filled
tenor	int	Amount (in seconds) until the maturity is reached
rateMantissa	int	Percentage of interest rate the borrower expects to return
depositWindow	int	Amount (in seconds) until the deposit maturity date is reached
maturityDate	date	Is established after the first transaction from currentDate + tenor. Timestamp
depositMaturity	date	Is established after the first transaction from currentDate + depositWindow. Timestamp
borrower	string	Pool borrower address
lastPaidTimestamp	int	Used to track the last timestamp at which an interest repayment for made
defaultedAt	int	Timestamp when the pool was marked as Defaulted

# Repayment

## Bullet Repayment Pools

Total debt for Bullet Repayment Pools consists of the sum of principal and interest accrued over the course of pool tenor. Interest or principal can't be repaid separately. Regardless of time of repayment, borrowers must repay the debt for the entire pool tenor.

Failing to repay pool or individual lender debt prior to maturity date, borrowers will have another 3 days (72 hours) to repay while accruing extra interest penalty.

## Monthly Repayment Pools

Interest repayment is an obligation to all lenders, therefore functionality to repay interest to individual lenders is not available. Borrower repays total interest accrued for the 30 day period, and is proportionally distributed to each lender based on their position size and duration in the pool.

Borrowers can repay monthly interest at any time within the 30 day period, however, If they fail to repay the monthly interest by the 30th Day, they then have 3 days (72 hours) to repay while paying an extra interest penalty.

# Lending

Lending to a Pool is restricted to whitelisted users. To whitelist lenders, the borrower must execute the **whitelist()** function within Pool smart-contract. The function will create a record in the **whiteList** map with the selected lender wallet addresses.

Lending is achieved through the **lend()** function which also requires **allowance()** over the assets. The Pool smart-contract will not hold custody of the loaned assets. Instead, the contract will immediately transfer the assets to the contract owner (the borrower wallet address) creating a record in the poolMembers map with **totalInterest**, **principal** and **timestamp**.

Upon every lend transaction, the **currentSize** is updated. Therefore, lending will be allowed until the **currentSize < maxSize** or **depositMaturity** is reached.

# Risks & Default

## In the event of missed payments of principal or interest

Borrowers that fail to pay principal and/or interest per the agreed due date will be provided 72 hours to settle all outstanding payments with the lenders:

- 1) During the 72 hours, an additional 5% in penalty interest over the pool's interest rate will accrue to the lenders on any unpaid principal or interest. Protocol revenue will not be collected on penalty interest.

- 2) Once the 72-hour period has elapsed, the lenders will have the right but not the obligation to re-classify the loan from “Overdue” to “Default”. This flexibility gives the lender leeway to continue collecting payments from the borrower, and for the borrowing to continue paying installments, without triggering the default process.
- 3) After the 72-hour period, the 5% penalty interest will continue to accrue on any unpaid principal or interest owed.

If a lender changes the loan classification from “Overdue” to “Default” as described above, all loans in the pool will be classified as such. Upon classification of a loan to “Default”, interest will stop accruing, and the legal process begins.

Once the default process has been triggered, the pool can be removed from the protocol and the lenders, as legal creditors, may pursue the borrower according to terms of the lending agreement.

The above procedures apply to loans with both bullet repayments and monthly repayments.

## In the event of borrower declaration of bankruptcy or default on other loans

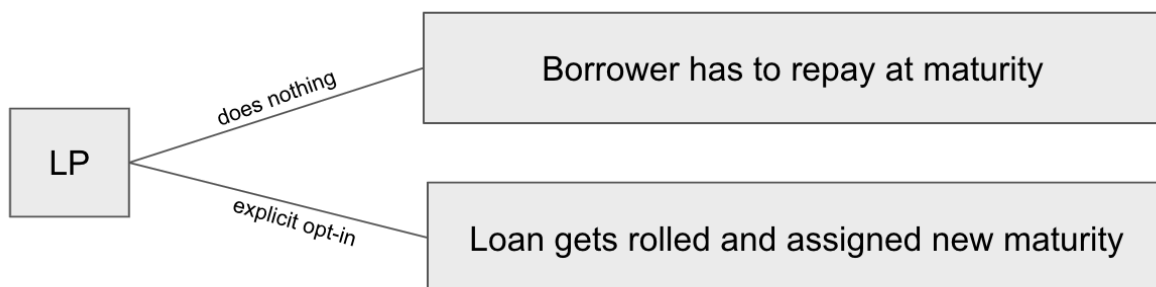
There may be cases in which a borrower declares bankruptcy or defaults on other loans issued outside the Clearpool Prime protocol (e.g., Alameda’s default in November 2022). In such cases, Clearpool will have the right to change the classification for all outstanding loans issued to the borrower from “Active” to “Default,” even if the loans have not matured or the borrower is in good standing.

## Roll

Up till forty-eight (48) hours before maturity, the borrower can request Rolling function.

The rolling functionality is only available for pools with 1 active lender. Rolling enables the extension of the loan duration for the single lender who explicitly accepted rolling. Lender who doesn’t not wish to roll may skip the request and the borrower will have to repay his debt.

When the **roll()** function is accepted, the pool maturity date will be updated accordingly. Each roll duration will be the same as the original loan tenor.



## Call-back order

After a lender has supplied liquidity to a Pool, they may request a call-back for early withdrawal. However, requesting withdrawal before the maturity date does not create an obligation for the borrower to repay. If the borrower opts to repay, interest will only be due on the period of time (timestamp delta) that have elapsed since the lend transaction was registered.

## Protocol fees

Calculations/Model here: [📄 Prime - Outstanding Items](#)

Protocol supports two types of fees: the **Origination Fee** and the **Interest Rate Spread**.

### Interest Rate Spread

Given:

$$interestAccrued = loanAmount * borrowApr * tenureYears$$

*spreadFactor* is a governance approved percentage of pool interest diverted to the treasury at the time of maturity/repayment of loan. Initially this amount is set by the Governor, but the value can be modified in the future. The Interest Rate *spread* is applied only on the “*repay*” transaction initiated by a borrower and charged to the lender. Interest rate spread is calculated on an annualized year with **360** days, and each standard month is **30** days.

$$spreadAmount = spreadFactor * interestAccrued$$

Where *spreadFactor* is adjustable by the Governor. If at any time the *spreadFactor* is adjusted a new value is applied **only** to pools created after the adjustment is made; old pools maintain the same *spreadFactor* value.

$$spreadAmount = spreadFactor * (loanAmount * borrowAPR * secondsElapsed/secondsPerYear)$$

**\*\*However**, the accrued penalty interest is not subject to interest rate spread charges. Penalty interest will start accruing immediately after missing the date and time of repayment.

### Origination Fee

The *originationFeeAmount* is directly sent to Clearpool Treasury when a borrower repay transaction occurs (final repayment including principal) and is charged to the borrower. Origination fee calculated on an annualized year with **360** days, and each standard month is **30** days.


This fee is calculated as an annualized percentage on the amount raised by a loan:

$originationFeeAmount = originationFeeFactor * secondsInTenure / secondsPerYear * loanAmount$

\* if a loan is repaid via callback, *secondsInTenure* = seconds elapsed up til the repayment timestamp.

\* If a loan is repaid without callback, *secondsInTenure* = duration of tenure of the pool in seconds

Where *originationFeeFactor* is adjustable by the Governor. If the fee is adjusted a new value is applied to newly created pools; old pools maintain the same fee value.

In the event that the pool / loan is rolled, protocol will charge a smaller origination fee for every roll (*rollingIncrement*) equal to x% of the initial *originationFeeAmount*. That means that the *originationFeeAmount* is applied only on the original tenure set on the pool, and an additional X% annualized added to the *originationFeeAmount* for every roll. [view example calculations here](#).  Prime - Outstanding Items

*rollingOriginationFeeAmount*

$= originationFeeAmount + ((rollingIncrement * originationFeeAmount) * (secondsElapsed / secondsInTenure))$

## Treasury Wallet Addresses

**eth:0x455011f2704c6E192b09d9CC1430299C70af3454**

**matic:0xDBB6fb1279d5592693695c150ed29a78eb62fA54**