The Minterest Lending Protocol
Technical Paper v1.4

September 2023

Introduction

Minterest is a decentralised lending protocol with a unique economic model. It captures more fee value than any other lending protocol and redistributes it as rewards to users who contribute to its governance with the highest long term yields. Minterest therefore optimises its utility for those who create its value - its community of users.

Feature Innovations

At its core, the Minterest protocol facilitates decentralised token markets for the pooled supply and borrowing of assets, with interest rates algorithmically determined by their demand dynamics. Emissions rewards in the form of Minterest's native MINTY governance token are issued to incentivise and reward users to create value through their various interactions.

Minterest's feature innovations revolve around breakthroughs in protocol managed auto-liquidation events, buyback mechanisms incentivising users to contribute to governance, and dynamically weighted voting rights delivering a 'loyalty reward' for long term participation.

A key Minterest innovation is the solvency of under-collateralised borrower positions being managed through auto-liquidation processes, where the protocol itself acts as the liquidator. This ensures the solvency of each user's position, and

the underlying asset pool, are maintained while liquidation fees contributed by insolvent borrowers are captured by the protocol for the benefit of its governing user community.

To participate in governance users stake MINTY in the protocol, with voting entitlement dependent on the quantity and length of time MINTY is staked. Users may unstake at any time but forfeit their accumulated voting weight doing so. The distribution of Governance Rewards, earned for staking and contributing to governance, is determined by a user's voting weight, resulting in a loyalty reward for long term participation. Users must vote or delegate on proposals a minimum once every 6 months, else their MINTY is forcibly unstaked, and any voting weight also forfeited.

Core Protocol Functions

The core functionality of the Minterest protocol is as follows.

Interest Rate Calculation

Interest rate calculation is separate for each token market and is based on the token market's Utilisation Ratio, detailed below. Interest rate calculations are applied to the pool of suppliers and borrowers in each token market and are adjusted dynamically as the supply and borrow relationship varies.

Interest rates in each token market are updated each time a transaction occurs, including but not limited to, supplying, borrowing, repaying, withdrawing or liquidation of borrowed assets.

Utilisation Factor

Borrowing from the protocol is over-collateralised, with the maximum proportion being the Utilisation Factor. Utilisation Factors are specific for each underlying token pool and are applied for each token asset in supplying and borrowing pools.

Utilisation Ratio

1. User Level

The solvency of any borrower position is determined by the Utilisation Ratio, which is the ratio between the collateral and the loan amount. A Utilisation Ratio less than 1.0 indicates the position is insolvent, with the protocol liquidating the required portion of the loan and collateralised assets to return the borrower's portfolio to solvency.

2. Pool Level

The Pool Utilisation Ratio is calculated for each individual pool and is not related to Utilisation Limits applied at the user level. The Utilisation Ratio **U** for each market unifies supply and demand into a single variable:

$$U_a = poolBorrowUnderlying_a / (poolSupplyUnderlying_a + poolBorrowUnderlying_a - poolProtocolInterest_a)$$

where:

poolBorrowUnderlying$_a$ - total value expressed in the underlying asset of outstanding borrows

poolSupplyUnderlying$_a$ - total value expressed in the underlying asset placed in the token pool

poolProtocolInterest$_a$ - total value expressed in the underlying asset of protocol interest accumulated in the underlying assets held in this token pool

The Utilisation Ratio is influenced by user operations i.e. every supply or borrow transaction causes the ratio to be recalculated and thus updates the Borrow Interest Rate. All fluctuations of the Borrow Interest Rate are accumulated using the Borrow Interest Rate Index. The interest a user pays is calculated using a lazy index

methodology based on two values, the current value of the index and the value of the index stored when the loan was taken.

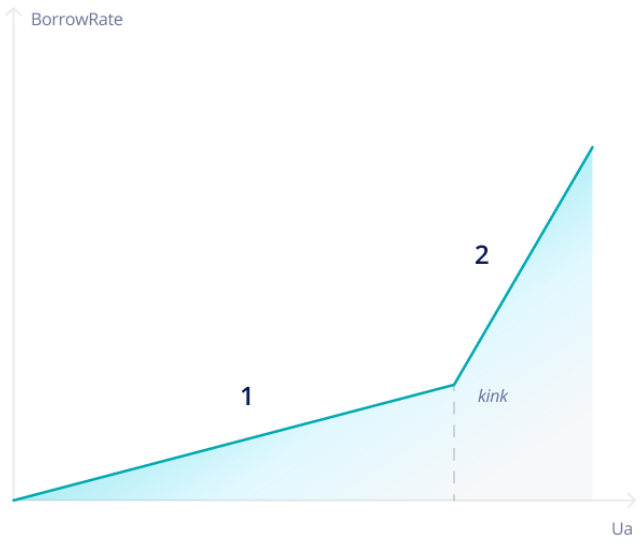
Calculating Borrow Interest Rate

The demand curve in each token market interest rate model is expressed as a function of its Utilisation Ratio.

Interest Rate Model

Token markets with token assets subject to price volatility utilise a kink interest rate model, where interest rates exhibit some form of kink and sharply increase at a defined threshold, the kink point. For efficiency, token markets with stablecoins in which a linear interest rate model is utilised simply have the kink point set at 100%.

The kink model contains two segments; pre and post the kink point. When the Utilisation Ratio reaches the kink point, an additional multiplier is applied to significantly increase interest rates.



For such models, when token pools experience significant borrowing demand in proportion to supply (where $U_a > kink_a$) such that the utlisation ratio is above the kink point, the borrowing rate increases steeply upward to dynamically incentivise

supply while discouraging borrowing, thus introducing demand and supply mechanics which support a sustainable equilibrium in the token market.

1. **U_a <= $kink_a$**

$$borrowInterestRate_a = U_a * interestRateMultiplier_a + InitialRate_a$$

where:

U$_a$ is the Utilisation Ratio for pool *a*

kink$_a$ the point at which the ***borrowRate***$_a$ formula increases steeply upwards

interestRateMultiplier$_a$ is the multiplier of utilisation ratio that determines the rate of increase of the interest rate

initialRate$_a$ is the base interest rate at the y-intercept when utilisation ratio = 0

2. **U_a > $kink_a$**

$$borrowRate_a = (U_a - kink_a) * kinkCurveMultiplier_a + (kink_a * interestRateMultiplier_a) + initialRate_a$$

where:

kinkCurveMultiplier$_a$ is the *multiplierPerBlock* post a kink point

Calculating Supply Interest Rate

The Supply Interest Rate earned by suppliers is implicit, and is equal to the borrowing interest rate, multiplied by the utilisation ratio of the token market..

$$supplyInterestRate_a = U_a * borrowInterestRate_a * (1 - protocolInterestFactor_a)$$

where:

U$_a$ - Utilisation Ratio for token pool *a*

borrowInterestRate$_a$ - the current per-block borrow interest rate for token pool *a*

protocolInterestFactor$_a$ - defines the portion of borrower interest that is converted into **protocolInterest**$_a$

Supplying

When users participate in the Minterest protocol and supply assets, they access the interface and deposit their token assets via signed transactions. From a user's perspective this occurs as the following.

A user supplying 10 ETH into Minterest's ETH pool will receive a receipt for 10 mETH tokens. As the first operation in the pool within the block, the protocol flow will be:

- The protocol initiates the accrual of interest on the ETH market.
- The ETH underlying asset pool is updated with the addition of 10 ETH.
- The user's balance is updated with the addition of mETH as per the current Exchange Rate.
- All other token pools are unaffected by this transaction.
- Other transactions in the current block affecting this pool do not trigger the accrual of interest but do trigger recalculation of interest rates in this pool.

The Exchange Rate is the variable which renders the fluctuation of Borrow Interest Rate into actual interest earned by a user. Since the Supply Interest Rate is implicit and is calculated based on the Borrow Interest Rate, the Exchange Rate through calculation of *marketBorrowUnderlying* allows the calculation of supply interest.

The Exchange Rate is calculated based on this formula:

$$ExchangeRate_a = (marketSupplyUnderlying_a + marketBorrowUnderlying_a - marketProtocolInterest_a) / marketSupplyWrap_a$$

where:

marketSupplyUnderlying$_a$ is the total value of underlying assets in the pool *a*

marketBorrowUnderlying$_a$ is the total value of all loans and interest borrowers will return to pool *a*

marketProtocolInterest$_a$ is the protocol fee in underlying assets accumulated by the market

marketSupplyWrap$_a$ is the total amount of receipt tokens on the market *a*

Borrowing

When borrowing a specified sum, the borrower supplies the required amount of collateral, and the protocol releases tokens from the pool of the borrowed underlying asset. When a borrower repays the debt, the protocol returns the tokens from the borrower to the pool of the borrowed underlying asset and updates their Utilisation Ratio.

Initial Borrow Flow

For a user who has borrowed 10 ETH and supplied the required collateral the protocol flow will be:

- Accrue interest and calculate Borrow Index.
- Update the **Total Borrowed** position for the user (0 as this is the first loan).
- Calculate New Total Borrowed = TotalBorrowed + 10 ETH.
- Assign current Borrow Interest Rate Index to user Interest Index.
- Store Interest Index, Borrow Interest Rate Index and User Total Borrowed.

Each time a transaction occurs, the **Borrow Interest Rate Index** for the asset is updated to compound the interest since the prior index, using the interest for the period, denominated by **borrowRate * blockDelta**, and calculated using a per-block interest rate:

$$borrowInterestRateIndex_{a,n} = borrowInterestRateIndex_{a,(n-1)} * (1 + borrowInterestRate_{a} * blockDelta_{a})$$

where:

borrowInterestRateIndex$_{a,(n-1)}$ is the accumulator of the total earned interest rate for token market *a* in the block number interest was last accrued in

borrowInterestRate$_{a}$ is the current per-block borrow interest rate for this token pool

blockDelta$_{a}$ is the number of blocks elapsed since the last accrual

Additional Borrow Flow

In a case where the user seeks to borrow an additional 20 ETH in a period 30 blocks later the flow will be:

- Accrue Interest
- Get current **Total Borrowed** for the user (10 ETH)
- Get current user **Interest Index** (Borrow Index 30 blocks ago)
- Calculate new user **Total Borrowed** (includes accrued interest for the 30 blocks) – 20 + (BorrowInterestRateIndex * TotalBorrowed) / InterestIndex
- Assign user Interest Index as current Borrow Interest Rate Index
- Store new user Total Borrowed and new Interest Index

Repayment and Redemption

When repaying a loan, the user transfers underlying assets into a market liquidity pool. The protocol accrues interest and calculates the loan share to be discarded from the user account.

As an example, if a user wishes to borrow in USDC using ETH as collateral for which the Utilisation Limit for the ETH pool is 80%, it means they can borrow up to 80 USDC against $100 worth of ETH collateral.

The borrower supplies $100 worth of ETH as collateral and borrows 60 USDC. In this scenario, the borrower's Utilisation Ratio will be 80/60 = 1.33. If the borrower then repays half the borrowed amount by returning 30 USDC, this is re-applied to the underlying ETH asset pool, which increases the Utilisation Ratio for the position to 80/30 = 2.66.

In the event the market works against the borrower where a drop in value of the collateral asset or an increase in the value of the borrowed asset results in an insolvent position, the protocol resolves this with an automated liquidation

procedure. It exchanges a proportion of the collateral on market, priced at a predetermined discount which acts as the protocol's liquidation fee, for the borrowed asset. This asset is then repaid to the underlying token pool from which it was borrowed, bringing the user's position back to solvency.

The withdraw function allows a user to convert their wrap receipt tokens into underlying assets using the Exchange Rate, which returns the underlying assets from the pool to their wallet.

Auto-Liquidation

The Minterest protocol undertakes the liquidator role, removing the need for external third-party liquidators. Minterest's automated liquidation process directly benefits borrowers, given algorithms do not require economic incentives to participate in liquidation processes. The protocol is able to economically liquidate smaller total percentages of borrower collateral than otherwise since the protocol only liquidates the amount required to return the borrower's collateral position back to solvency.

The protocol compares the USD value of all borrower collateral positions against their Utilisation Ratio, with the USD value of all loans including accrued interest on supplying and borrowing. If the Utilisation Ratio is less than 1.0, the borrower's portfolio is insolvent and so subject to liquidation.

As detailed in the Minterest Whitepaper, the protocol flow for Solvency Engine is:

- An automated bot array scans protocol solvency, replacing the external liquidator function in identifying under-collateralised borrower positions. Each bot database mirrors the chain, with portfolio provisioning ensuring security and redundancy.
- Once an under-collateralised position is identified, the amount of collateral required to be sold and the most efficient sell pair are both algorithmically determined.

- The algorithm works with DEXes to define the best flash loan provider, as well as optimal swap routes to exchange the seized collateral asset and repay the taken flash loan.
- The off-chain sequence results in a packed call data, sent to the chain.
- The smart contract executes the transactions packed into the call data: take a flash loan, repay the insolvent loan, seize the collateral, swap the collateral into the borrowed asset, repay the flash loan and also swap the surplus into a stablecoin.
- If the smart contract determines any precondition or calculation to be invalid, the liquidation event is cancelled.
- Upon confirmation, the borrow position is again solvent and the liquidation event is complete.

A liquidation event for one loan position always affects only one collateral asset and happens within one flash loan transaction. In some cases this may be not enough to get a user to solvency, so a series of liquidations affecting different collateral and borrowed assets is possible.

Emissions Rewards Distribution

Standard Rewards are distributed to both suppliers and borrowers. Individual user allocation is dependent on the proportional share in each supply or borrow market. Allocations of individual token markets can be varied determined by governance, which allows optimisation of the protocol's economy. This allows for the relative proportions of Standard Rewards apportioned to token markets from the existing daily rate, or the Standard Rewards daily rate, to be varied to achieve the same outcomes.

The emission rate is set per token market per action for both supply and borrow and defines the number of MINTY distributed per block e.g. an emission rate of 10 on supply means all liquidity suppliers will receive 10 MINTY per block in total.

MINTY Buyback Flow

The Minterest protocol accumulates tokens through the activity of its value capture functions, with such value being accumulated as stablecoins. The protocol periodically swaps accumulated tokens on-market for MINTY tokens for distribution to its users.

The Buyback flow includes the following:

Accumulation of Fees

The protocol accumulates fees in the form of tokens in its various pool reserves. The tokens are intermittently swapped for MINTY to be stored in the protocol's reserves for distribution as Governance Rewards. Non-stablecoins accumulated in the protocol's reserves following liquidation events are swapped for stablecoins to avoid price volatility. Assets accumulated from managing token markets and supplying flash loan liquidity are stored in their underlying pools, and if required also swapped for stablecoins.

Exchanging Assets for MINTY Tokens

Stablecoins accumulated in the Preliminary Treasury are swapped for MINTY tokens on-market with these exchange events reducing holdings of stablecoins and accumulating MINTY tokens ready for distribution to users.

Distribution

The protocol triggers Governance Rewards distribution events on a recurring basis. During each event the protocol distributes the appropriate portion of each user's MINTY Governance rewards to them. This forms a 'dripping sequence' of rewards

occurring on a near hourly basis instead of a single large distribution, preventing users from gaming the timing of their protocol interactions.

The proportion applied to the buyback can be varied by governance and will be influenced by the current and predicted effectiveness of the protocol in executing its value capture mechanisms, as well as historically accumulated balances.

Governance Rewards Distribution

Governance Rewards are distributed proportionally to users who stake and participate in the protocol governance in accordance with their buyback weight, with buyback weight determined by:

- the amount of MINTY staked in the protocol, including
    - Manually staked MINTY
    - Earned Standard Rewards
    - Previously earned Governance Rewards
    - MINTY eligible for withdrawal from the Vesting contract
- the period of time MINTY is staked.

The mathematical representation for the rewards distribution is:

$$R = (X + PriorRemainder) * Y\%$$

where:

R is the rewards amount to be distributed to all users

X is the amount of fees accumulated in a month

Y is the portion of **X** fees to be distributed the following month

PriorRemainder is the remaining prior-month balance before the Y% is taken

The R distribution is delivered in an hourly drip over the following month at R/730 per hour, which is **(X + PriorRemainder) * Y% / 730** per hour i.e. run every N minutes where N=60, for an hourly drip.

Voting and Buyback Calculations

MINTY staked in the protocol, enabling governance participation, determines a user's voting weight and buyback weight, with each employing separate methods in their calculation.

Buyback Weight

Illustrating buyback weight with an example, Lenny has staked 100 MINTY in governance. He also has another 100 MINTY subject to a 6 month cliff, with a 12 month unlock schedule after. Lenny also earned a total of 100 MINTY from a combination of Standard Rewards and Governance Rewards two months prior, which he then manually staked. Lenny therefore benefits from the protocol's loyalty reward with the additional application of a two month loyalty factor.

The formula for Lenny's buyback weight is:

$$BuybackWeight = (100+100/12*2)*1.02 =119$$

where

Loyalty factor is 2%.

Claimable unlock schedule value is 0

Manually staked MINTY is 100

Claimable rewards is (2/12 of 100)

Lenny's earned loyalty reward is therefore 2.33 MINTY

Voting Weight

The formula for Lenny's voting weight before applying the loyalty reward is:

$$VotingWeight = 100/12*2+100+100 +2.33 =218.99$$

where:

MINTY unlock schedule value is 100/12*2

Manually staked MINTY is 100

Earned emissions is 100

Loyalty reward is 2.33

In calculating voting weight, buyback weight is added to a user's voting weight.

Voting Weight and Loyalty Reward

The proportion of Governance Rewards distributed to each user is determined by their buyback weight. Buyback weight is determined by the total amount of staked claimable MINTY, multiplied by a user's loyalty factor. This factor increases with time, so those who stake in governance longer earn higher rewards.

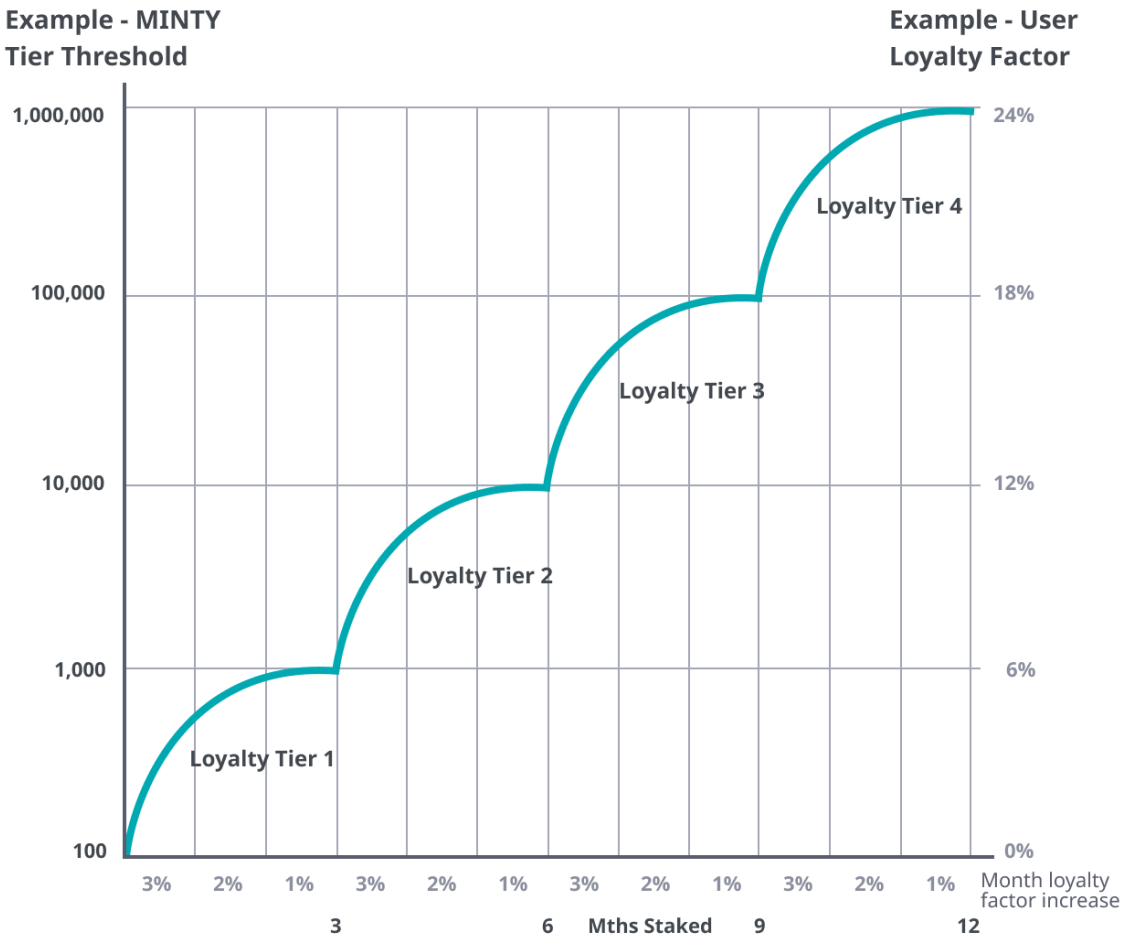
Loyalty Factor

Loyalty factor is a multiplier applied to a user's total claimable MINTY, disregarding when each token was staked. This results in a user's buyback weight increasing due only to increasing loyalty factor, and thus without the need for additional staking or rewards accrual.

The ability of a user's loyalty factor to continue to increase is limited by the requirements of sequentially increasing loyalty tiers. The maximum loyalty tier a user is able to be assigned to is determined by their MINTY balance and length of time staked.

Unstaking, and whether such actions break or are within the conditions set by the loyalty tiers, also influences the movement of a user's loyalty factor up and down the tiers.

Example of Loyalty Tiers



The increase in loyalty factor occurs within multiple time periods within a calendar month and is subject to logarithmic functions, and so is non-linear. Time periods earlier within a tier have sharper rates of increase in loyalty factor than later time periods.

In the example above, the loyalty factor in each tier increases by 3% in the first month, 2% in the second month and 1% in the last month.

Users benefit from a near immediate significant impact in their loyalty factor at the start of any loyalty tier, positively impacting their future Governance Rewards earnings. This acts as a supportive incentive to continue staking and participating in governance. As the user moves up their loyalty tier in later periods these increases in loyalty factor flatten. A new incentive occurs, namely the impending sharp rate of increase in loyalty factor provided by the next loyalty tier.

An example illustrates this.

Loyalty Tier 1 may have a range from a minimum 100 MINTY to 1000 MINTY and contain 3 loyalty factors applied over 3 months i.e. 3%, 2% and 1%. A with less than 100 MINTY will not receive any loyalty factor, while users with a staked MINTY balance within the range can receive a maximum 6% loyalty factor within three months.

To further increase their loyalty factor, their required staked MINTY balance needs to exceed 1000 MINTY, by staking or earning additional MINTY. Upon doing so the user moves into Loyalty Tier 2 where they can receive a higher loyalty factor by continuing to stake over the following 3 months. If they meet the minimum floor conditions of MINTY staked required for Loyalty Tier 3 they can then advance up it with the same applying for Loyalty Tier 4.

Loyalty Core and Unstaking Sanctions

While users may unstake all of their claimable MINTY balance at any time, unstaking more than than a percentage preset by the protocol subjects them to a switching cost, where sanctions are applied to their loyalty factor should they do so.

Loyalty Core

The protocol tracks a user's staked MINTY tokens to calculate a loyalty core, the minimum amount of MINTY required kept staked to preserve their loyalty factor. User's unstaking more MINTY than required to maintain their loyalty core have sanctions applied to their loyalty factor. This results in them being moved by the protocol to lower loyalty tiers, decreasing their loyalty factor, and thus their loyalty reward. Additionally, the protocol tracks the natural movement of users between loyalty tiers adjusting their loyalty factor as they do.

Unstaking Sanctions

As an example using the illustration above, Lenny has staked 110,000 MINTY (disregarding their source) more than 12 months ago and is in Loyalty TIer 4. If the maximum percentage of MINTY allowed to be unstaked in a given month is set at 10%, Lenny may withdraw up to 11,000 MINTY tokens, and his loyalty core is therefore 99,000 MINTY.

Let's say the sanction for breaching the loyalty core is set as a reduction of one loyalty tier and loyalty Tier 4 has a lower limit equal to 100,000 MINTY.

Lenny unstaking his MINTY has three possible outcomes.

1. Maintaining the loyalty core

He unstakes less than 10,000 MINTY so unstaking does not break the loyalty core or the lower limit amount of MINTY required to maintain his loyalty factor. Lenny therefore stays within Tier 4.

2. Breaking the tier lower limit

Lenny unstakes 11,000 MINTY, with his new balance of 99,000 MINTY breaking the lower limit required to maintain his Tier 4 loyalty factor but not the loyalty core of 10%. The protocol naturally reduces him to Tier 3, limiting him to the maximum loyalty factor of Tier 3.

3. Breaking both the loyalty core and tier lower limit

Lenny unstakes 15,000 MINTY, more than the required 11,000 to maintain his loyalty core, which naturally lowers him to Tier 3. However the protocol applies a sanction to Lenny for breaching the 10% loyalty core, given he unstaked more MINTY than allowed in a given month and lowers him a further loyalty tier to Tier 2. Lenny is still able to reach a maximum loyalty factor of Tier 3, but can only do so over time as new increases in his loyalty factor are applied.

Governance Rewards

The exact proportion of Governance Rewards a user receives is calculated using their buyback weight. The protocol utilises a lazy indexing methodology, with the index defining rewards able to be claimed per unit of buyback weight.

$$currentIndex = oldIndex + (dripAmount/totalWeight)$$

where:

dripAmount is the sum of MINTY being distributed every hour

TotalWeight is the total amount of MINTY user holds in the protocol with regards to their loyalty factor

Using the delta between the last time when the user claimed MINTY from Governance Rewards and the current index, calculating current earned value is:

$$earnedValue = (currentIndex - userStoredIndex) * currentUserWeight$$

where:

currentUserWeight is the total amount of MINTY staked in the protocol

userStoredIndex is the index value as of the last withdrawal

Emission Vesting Logic

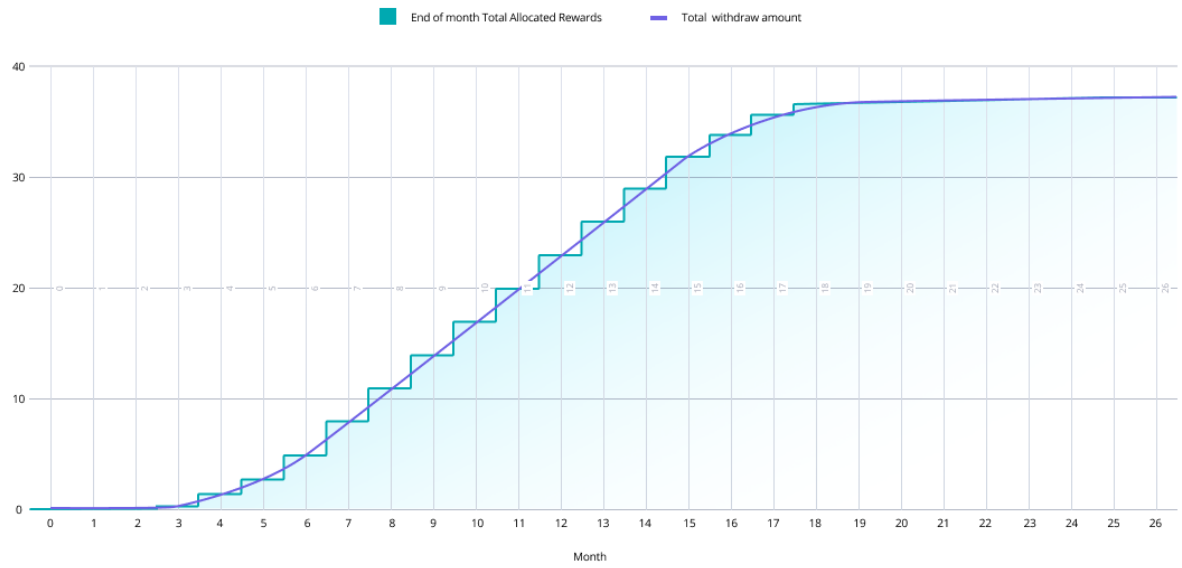
All emission rewards vest every token over 12 months from when it was allocated to a user with the distribution to be calculated through the application of delay logic. Given it is technically unfeasible to track every token state on-chain, a specific logic is applied to achieve the same result, resulting also in significantly reduced protocol gas expense.

The delay logic methodology revolves around 12 periods of calculation. Each period represents approximately 1/12 of a year, or 30.4 days. The contract accumulates the rewards of a single user for the 30.4 days into a single value with the accumulated amount allocated to the period. For any moment of time the withdrawable amount of MINTY tokens for a specified period of time can be calculated as follows:

$$segmentWithdrawAmount = slot1 + slot2/12*11+slot3/12*10...+slot12/12$$

At the end of each period the protocol defines segmentWithdrawAmount for the next 30.4 days and vests this amount on a per block basis. In case of user inactivity all rewards are distributed equally between the segments of inactivity. If the inactivity period is longer than 12 months, rewards for the segments older than 12 months have become 100% vested and so fully available for withdrawal.

Graphically, the user receives a smooth vesting curve, meaning they experience a smooth vesting schedule. The chart below depicts the steps of recalculation and the actual distribution curve for an active user.



The algorithm works differently for a non-active user. It utilises averaging under the hood, as a result if a user has no activity for longer than a month the total amount available for withdrawal drops. This happens after every next month of inactivity, so the chart resembles a chain saw pattern.

Minterest NFTs

Minterest NFTs have functional utility, meaning they enable the holder to interact with the protocol in ways that deliver benefits through their ownership. Each NFT is unique and has specific benefits provided to its holder.

An NFT smart contract allows minting of the NFTs based on unique illustrations designed by a portfolio of prominent creative artists. Each image is assigned a unique immutable identifier shared by other NFTs of that type.

Each NFT belongs to a specific tier with each tier granting a distribution of NFT Rewards equivalent to a percentage of the user's Standard Rewards, referred to as

its boost. The NFT boost is fixed and applies to every market the user participates in for both supply and borrow activity.

The exact boost size of each tier of NFT is calculated off-chain. The proportions of the total emissions of Standard and NFT Rewards are split 66.7% (⅔) to 33.3% (⅓) respectively. This approach grants predictability of the total emission value for the period in which the NFT is valid.

As an example, if User 1 is the only liquidity provider in all markets who also holds the top tier NFT Satoshi Nakamoto with a 50% boost, they will receive exactly 100% of the emissions rate from combined Standard and NFT Reward distributions.

If the emission rate is 100 MINTY per block, given User 1 has 100% of the market and a 50% boost NFT, User 1 will receive rewards of 66.6 MINTY Standard Rewards and 33.3 MINTY NFT Rewards. However, 100% of emissions being claimed is an unrealistic edge case, since total supply is never controlled by a single user.

NFT Rewards

NFT Rewards of 2.6M MINTY represents the maximum allocation that is possible given the total Standard Rewards pool is 7.8M MINTY. This ensures no NFT user can impact the value of Standard Rewards of any other user, whether they are an NFT holder or not.

NFT Rewards are issued in proportion to Standard Rewards i.e. an NFT with a 30% boost has the owner receive NFT Rewards equivalent to a 30% increase in their Standard Rewards.

As the proportion of liquidity supplied by users without NFTs increases over time the quantum of MINTY issued as NFT Rewards declines. The maximum allocation of NFT Rewards is possible only if all liquidity supplied in all pools is supplied by the holder of the Satoshi Nakamoto NFT and its 50% boost, an unrealistic edge case.

The following example of 3 users with NFTs illustrates this:

- User 1 has 50% of the market with 50% boost and so receives 33.3 MINTY Standard Rewards and 16.65 MINTY NFT Rewards.
- User 2 has 30% of the market with 20% boost and so receives 19.98 MINTY from Standard Rewards and 3.99 MINTY from NFT Rewards.
- User 3 has 20% of the market with 40% boost and so receives 13.32 MINTY Standard Rewards and 5.3 MINTY NFT Rewards.

In this scenario the total Standard Rewards amount remains at 66.6 MINTY but NFT Rewards decrease to 25.85 MINTY. Total emissions are now less than 100 MINTY per block, instead being 92.45 MINTY, creating a surplus of 7.55 MINTY.

Additionally, NFTs have periods of validity ranging from 5 months to 3 years and eight months so as they expire the NFT Rewards apportioned to them becomes zero. Both these events mean a surplus of NFT Rewards develops immediately after the protocol's public launch, which once crystalised is distributed to the Strategic Reserve. The same principle also applies to Buyback Rewards, where unused MINTY tokens allocated to the pool join the Strategic Reserve, increasing its balance over time.

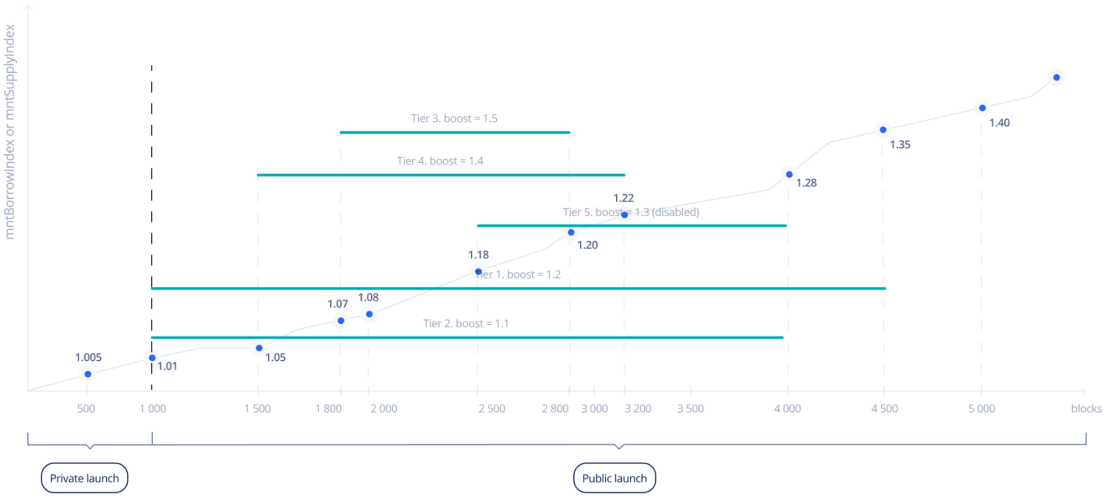
NFT Rewards Boosts

NFT Rewards are not indefinite and do expire. The period of validity known as time-to-live **(TTL)** is a set of timestamps which determine when the NFT Rewards functionality begins and ends. The NFT is tradeable, and to track ownership change user state information is updated on transfer of ownership of each NFT.

NFT boosts are not stackable. In cases when a user has two or more NFTs the protocol applies only the largest boost. When a NFT's TTL expires, the protocol switches to the highest boost of any other active NFT held by the user's wallet.

To reduce calculation expense, all NFTs minted during private launch initiate TTL together, defined as the timestamp when the protocol moves from private launch phase to public launch. A whitelist function of the NFT limits access to the protocol during its private launch phase to only those wallet addresses holding an NFT.

NFT Boost Calculations



This graph represents a specific user. The X-axis defines time parameters, the Y-axis **mntBorrowIndex** or **mntSupplyIndex** value. The horizontal lines indicate the emission boost from the NFT, the numbers above defining the **emissionBoost** value for a particular tier.

During the protocol's private launch phase, the NFTs function similarly to a VIP card, with access to the protocol exclusive to NFT holders. Boosts attributed to NFTs during this phase do not apply. When the protocol initiates public launch, the timestamp and required indices are created in the contract to launch NFT boost calculations.

To determine the emission boost for a specific user, the numbers of blocks for changing the maximum boost for a given user, as well as the values of the **mntSupplyIndex** and **mntBorrowIndex** parameters for these blocks, are calculated. In the graph above these are blocks 1500, 1800, 2800, etc.

Where a user already holds an NFT, the **emissionBoost** initiates based on its tier. The maximum boost value for the user changes where an NFT with maximum **emissionBoost** expires or a user sends or receives an NFT changing the value of the maximum **emissionBoost** for that user.

In calculating the number of MINTY tokens earned by a user, taking into account boosts:

$$deltaAccrued = userWrapBalance * \Sigma(emissionBoost_i * (index_{i+1} - index_i))$$

The chart shows that the user had the following sequence of boosts:

1. Index values 0-1000 - no boost
2. Index values 1000-1500 - boost is 1.2
3. Index values 1500-1800 - boost is 1.4
4. Index values 1800-2800 - boost is 1.5
5. Index values 2800-3200 - boost is 1.4
6. Index values 3200-4000 - boost is 1.3
7. Index values 4000-4500 - boost is 1.2

The sequence illustrates the contracts tracking TTL of NFT tiers and where a user either transfers or acquires an NFT, the best boost available is applied.