The Equalis Protocol

Matthew L. Hooft

EqualFi Labs

E-mail: mhooft@equalfilabs.com (Matthew L. Hooft)

Abstract. We present EQUALIS, a decentralized financial protocol providing pooled credit, bilateral options-like agreements, and yield-bearing limit orders that enable spot trading, without oracle dependence for correctness. In contrast to liquidation based systems that couple borrower outcomes to continuous price movements and oracle updates, EQUALIS isolates risk to explicit user choices, time, and bilateral agreement structure. The protocol introduces transferable Position NFTs as account containers, a dual index accounting model for fee distribution and maintenance charging, and a direct agreement layer for cross asset credit that does not require price oracles. The direct layer includes both agreement-based term and rolling loans as well as lightweight limit orders that enable gas-efficient trading via direct asset transfers without agreement creation. Defaults are handled through bounded, rules based settlement rather than collateral liquidation cascades, and protocol losses are not socialized across depositors.

We describe the protocol architecture, state machines, and accounting invariants required to preserve pool solvency under deposits, withdrawals, borrows, repayments, and direct agreement flows. We provide informal proof sketches for key safety properties including pool isolation, deterministic encumbrance accounting, and non socialized loss. The protocol enables yield-bearing limit orders (YBLO) where escrowed funds remain yield-accruing in pools while encumbered, supporting both credit agreements with options-like outcomes and spot trading via direct asset transfers. Yield and incentives are deterministic: a fee index distributes pool fees, a maintenance index charges over time, and an Active Credit index distributes time-gated rewards to P2P lenders (on lent principal) and same-asset borrowers (on debt) with weighted dilution; cross-asset debt is excluded from Active Credit participation.

1 Introduction

Decentralized credit and exchange systems have largely converged on a single dominant pattern: over collateralized borrowing with liquidation based enforcement. In these systems, loans remain open ended obligations whose safety is maintained through continuous repricing of collateral and reactive liquidation when account health falls below a threshold. This pattern has proven workable, but it also imposes structural costs. Borrowers are exposed to abrupt loss of collateral during transient volatility, liquidity shocks can cascade through liquidation auctions, and protocol behavior depends on oracle availability and on market depth at the moment liquidation is triggered. These characteristics create an experience that resembles margin trading more than consumer credit, and they make the protocol's safety properties inseparable from external price dynamics.

This paper studies an alternative design point that we refer to as *non reactive finance*. The core idea is to minimize or remove mechanisms whose primary mode of safety is continuous reaction to price updates. Instead, we constrain outcomes through deterministic rules over time, explicit user chosen parameters, and localized accounting invariants. The goal is not to claim that price risk disappears, but to move how that risk is expressed away from liquidation cascades and toward bounded, legible settlement rules that are enforced without external market triggers.

We present EQUALIS, a decentralized financial protocol that provides three complementary primitives: pooled credit, bilateral options-like agreements, and spot trading via yield-bearing limit orders.

First, EQUALIS offers pooled single asset credit with Position NFTs that act as transferable account containers. Each position aggregates deposits, borrows, and encumbrances while remaining portable. This enables secondary market transfer of lending and borrowing states, while maintaining strict pool isolation. The pooled system uses a dual index accounting model. A fee index distributes system generated fees to holders as yield, and a maintenance index applies proportional charging over time. A normalized fee base is used to prevent recursive amplification of yield through internal feedback loops.

Second, EQUALIS includes DIRECT, a bilateral agreement layer that enables cross asset credit without price oracles and supports yield-bearing limit orders for spot trading. A lender and borrower negotiate an offer specifying principal, collateral, interest, and maturity. At acceptance, collateral is locked, principal is transferred, and interest is realized up front as a premium. The resulting agreement is settled deterministically through repayment, early borrower exercise, or a lender recovery path after a grace period. Additionally, the same offer mechanism supports spot trading through yield-bearing limit orders (YBLO) that execute direct asset transfers without creating agreement state, enabling gas-efficient trading while funds earn yield in pools. Crucially, the pooled system does not underwrite the credit risk of these agreements. Risk remains isolated to the two positions that enter the contract.

A central design choice of EQUALIS is how defaults are handled. Instead of liquidating collateral into the market, the protocol applies bounded, rules based settlement. The exact enforcement mechanism is defined at origination and triggered by time and state, not by reactive price thresholds. This approach is intended to avoid liquidation cascades and to make borrower downside legible at loan creation. The protocol further avoids socialized losses. Depositors are not implicitly backstopping other users through shared bad debt pools. When a position fails to meet its obligations, the settlement affects only that position's state and explicitly encumbered assets.

The system is implemented as a modular architecture with clear facet boundaries and explicit storage separation. We focus on correctness and security by stating invariants over balances, indices, and encumbrance variables, and by defining state machine constraints for pooled and direct flows.

1.1 Contributions

This paper makes the following contributions:

1. We formalize a non reactive financial design in which borrower outcomes are enforced by deterministic settlement rules rather than liquidation based collateral auctions.

2. We introduce Position NFTs as transferable account containers that unify deposits, borrows, and direct agreement state while preserving strict pool isolation and, for managed pools, whitelist gating by Position ID.

3. We describe a dual index accounting model with a normalized fee base, separating fee distribution and maintenance charging into two indices with explicit update rules, plus an Active Credit index that time-gates rewards for P2P lenders and same-asset borrowers; penalty flows also accrue to Active Credit.

4. We specify an oracle free bilateral agreement layer for cross asset credit, including offer posting, acceptance, repayment, early exercise, and recovery after a grace period.

5. We introduce yield-bearing limit orders (YBLO) as a lightweight trading mechanism that encumbers capital at specified price ratios while earning yield, enabling gas-efficient spot trading via direct asset transfers with partial fill support.

6. We demonstrate agent-friendly deterministic execution through executable offer books suitable for market makers and automated trading strategies.

7. We provide a set of accounting invariants and state machine constraints, with informal proof sketches for key safety properties including non socialized loss and deterministic encumbrance accounting.

1.2 Paper Organization

Section 2 defines the model, assumptions, and adversary. Section 3 provides a system overview and core invariants. Sections 4 and 5 specify pooled credit and direct agreements, respectively. Section 6 describes the associated index module and fee pot mechanism where relevant. Section 7 states correctness properties and provides proof sketches. Section 8 discusses security considerations and attack surfaces. Section 9 surveys related work, and Section 11 concludes with limitations and future directions.

2 Model, Assumptions, and Threat Model

This section defines the execution model, the security assumptions under which Equalis is intended to operate, and the adversarial capabilities considered in later analysis. The goal is to separate what the protocol *guarantees by construction* from what it can only guarantee under external assumptions about the chain, tokens, and user behavior.

2.1 Execution Model

Chain model. We assume an Ethereum compatible execution environment with atomic transactions, deterministic EVM semantics, and eventual finality. State updates are serialized according to the block producer's ordering. Adversaries may influence ordering through transaction submission strategies and through miner extractable value (MEV) tactics, but cannot violate EVM correctness.

Asset model. Pools and agreements operate over standard fungible tokens. Let $\mathcal{T}$ denote the set of supported ERC20 like assets. We assume a subset of tokens behave as expected with respect to transfers and balances, but we explicitly consider deviations in the threat model (Section 2.4.2). The protocol never assumes external price information for safety. Any economic value judgments are made by users when choosing offers and terms.

User model. Users control externally owned accounts (EOAs) and may deploy smart contracts that interact with the system. Users may create multiple positions and interact across multiple pools. Users may also behave adversarially, including self dealing between their own positions. This paper treats self dealing as allowed behavior unless it violates explicit invariants.

Position model. A *Position* is a protocol defined account container identified by a Position NFT. Each position holds accounting state across one or more pools. Positions are transferable, meaning ownership changes can transfer the entire state bundle. The protocol enforces pool scoped accounting such that a position's state in pool P_i does not contaminate another pool P_j. Managed pools may additionally gate membership via whitelists keyed to Position NFTs, while unmanaged pools remain permissionless.

2.2 Notation and State Variables

Let P denote a pool with underlying asset $\tau(P) \in \mathcal{T}$.

For a given position x and pool P, we use the following canonical variables:

- $\mathrm{dep}_{x,P}$: deposited principal credited to x in pool P.
- $\mathrm{debt}_{x,P}$: outstanding pooled borrow obligation of x in pool P.

- $\text{locked}_{x,P}$: principal locked as collateral or encumbrance in pool P and unavailable for withdrawal.
- $\text{offerEscrow}_{x,P}$: principal escrowed to fund posted direct offers in pool P.
- $\text{directLent}_{x,P}$: principal lent by x via active direct agreements in pool P.
- $\text{directBorrowed}_{x,P}$: principal borrowed by x via active direct agreements in pool P.

Pool level variables include:

- trackedBalance_P: internal accounting of the pool's token balance.
- I_P^{fee}: fee index for distributing system generated fees to positions.
- I_P^{mtn}: maintenance index for applying proportional maintenance charges.
- I_P^{act}: Active Credit index for time-gated rewards to P2P lenders and same-asset borrowers.
- $B_{x,P}^{\text{fee}}$: fee base used to compute position level fee accrual.
- $B_{x,P}^{\text{act}}$: Active Credit base contributed by position x in pool P (lent principal for lenders, same-asset debt for borrowers); cross-asset debt does not contribute.

We treat these as abstract variables in this section. Their concrete update rules are specified in later sections.

2.3 Assumptions

The following assumptions are required for the strongest claims in this paper.

No oracle dependence for safety. The protocol does not require external price oracles to maintain solvency of pooled balances. Cross asset credit is expressed only through bilateral direct agreements whose correctness does not rely on price feeds.

Token transfer sanity for supported assets. For tokens included in $\mathcal{T}$, we assume transfers either succeed and move the specified amount, or revert. Tokens with fee on transfer, rebasing semantics, callback hooks, or balance manipulation behavior may violate these assumptions and are discussed as threats.

Time source. We assume block timestamps are monotonic and within customary miner manipulation bounds. Maturity and grace period checks use block time and thus inherit typical timestamp uncertainty. We design timing thresholds to tolerate small manipulation without enabling profitable settlement deviations.

No privileged seizure beyond specified rules. Administrative control exists only for clearly scoped configuration and emergency procedures. The protocol does not rely on discretionary seizure or manual intervention for normal settlement. All enforcement is expressed as state machine rules.

2.4 Threat Model

We consider adversaries that attempt to steal funds, cause incorrect accounting, or force protocol level loss. The adversary may be a user, a contract controlled by a user, or an MEV searcher with ordering influence. We assume the adversary can create unlimited positions, post and accept offers, and interact across pools.

2.4.1 Adversarial Capabilities

Transaction ordering and MEV. An adversary may front run, back run, or sandwich transactions, and may attempt to exploit timing boundaries at maturity or grace period edges. The protocol must remain correct under any ordering of valid transactions. Flash loans enable same transaction state manipulation attempts, including index updates, fee base changes, and offer acceptance ordering. The protocol's invariants must hold even when an adversary can execute arbitrary logic between flash loan disbursement and repayment within a single atomic transaction.

Reentrancy. An adversary may attempt to reenter during token transfers or external calls. The protocol must be safe against reentrancy that could violate invariants, including cross facet reentrancy in a modular architecture. Flash loans introduce an explicit external callback boundary where the receiver contract executes arbitrary code before repayment. This callback can attempt to reenter any protocol function, including deposits, withdrawals, borrows, and offer acceptance. The protocol enforces diamond level non reentrancy guards to prevent state manipulation during flash loan execution.

Self dealing and looping. An adversary may create multiple positions and form agreements between them. This includes attempts to amplify fee accrual or to create circular lending structures. Such behaviors are permitted unless they create protocol level insolvency or violate explicit accounting constraints. The protocol should bound any amplification through conservation laws and fee base normalization.

Griefing and denial of service. An adversary may attempt to fill storage with offers, trigger worst case loops, or craft transactions that induce high gas usage. The protocol must avoid unbounded iteration and must prefer constant time operations where possible.

2.4.2 Token and Interface Hazards

ERC20 in the wild is not a law of nature, it is a suggestion. We explicitly consider the following hazards:

- **Fee on transfer and deflationary tokens:** transferred amount received differs from amount sent, breaking trackedBalance_P reconciliation.
- **Rebasing tokens:** balances change without transfers, invalidating balance based invariants and yield attribution.
- **Non standard return values:** tokens that do not return boolean success, or that return false without reverting.
- **Callback hooks:** tokens that can trigger external calls during transfers and approvals, amplifying reentrancy risk.

In this paper, safety claims are made for supported token sets that exclude these pathological behaviors, or require explicit adapter logic. Section 8 discusses mitigations and recommended token allowlisting policies.

2.4.3 Protocol Specific Threats

Incorrect encumbrance accounting. A primary protocol risk is inconsistency between deposited funds, locked collateral, escrowed offer funds, and active direct agreement principals. The threat is that a position could withdraw or reuse funds that should be encumbered, causing pool level insolvency. We address this by defining per pool encumbrance variables and enforcing them as hard constraints on withdrawals and new obligations.

Timing manipulation around settlement. Direct agreements depend on maturity and grace period checks. Adversaries may attempt to repay just after maturity, exercise at boundary times, or recover collateral at the earliest permitted block. The state machine is designed so that only one terminal path is valid at any time, and the permissible caller for each transition is explicit.

Cross pool contamination. Positions can span multiple pools. The protocol must prevent a user from creating obligations in pool P_i that implicitly depend on assets in P_j unless a direct agreement explicitly locks assets in the correct pool. We treat pool isolation as a first class invariant.

2.5 Security Goals

Given the model and threats above, the core security goals are:

1. **Accounting correctness:** pool balances and position balances remain consistent under arbitrary transaction ordering.
2. **No socialized loss:** one position's failure cannot directly reduce another position's deposited principal, aside from fees, explicitly agreed transfers, and bounded penalties whose distribution is predefined.
3. **Pool isolation:** assets in one pool cannot be implicitly seized or encumbered to satisfy obligations in another pool.
4. **Deterministic settlement:** direct agreements settle through a well defined state machine triggered by time and explicit calls, not by price thresholds.
5. **Bounded per call complexity:** core operations do not require unbounded iteration and remain feasible under typical gas limits.

The remainder of the paper specifies the mechanisms that implement these goals and provides invariants and proof sketches supporting their correctness.

3 System Overview

This section provides a top down description of Equalis as a set of interacting modules, and it introduces the core invariants that later sections formalize. The emphasis is on what state exists, where risk is located, and which mechanisms are intentionally absent.

3.1 Design Thesis

Equalis is designed around non reactive risk semantics. The protocol does not attempt to maintain safety by continuously repricing collateral or by liquidating users into the market when a price threshold is crossed. Instead, it constrains outcomes through deterministic settlement rules, time based transitions, and explicit encumbrance accounting. The system intentionally avoids utilization curves and other reactive rate models. Users select terms and rates by choosing pools and by entering direct agreements.

Two principles guide the design:

- **Risk localization.** Risk is local to the position and the pool where it is created. Cross asset risk is local to the two positions that sign a direct agreement.

- **Explicit encumbrance.** Any unit of principal that cannot be withdrawn is explicitly tracked as locked, escrowed, or otherwise encumbered. The protocol treats encumbrance variables as first class constraints, not as derived quantities.

3.2 Modules

Equalis comprises three primary components.

Pooled lending. Pools are single asset. A pool P accepts deposits of $\tau(P)$ and issues deterministic accounting claims to positions. Borrowing from the pool creates a pooled debt obligation for a position. Pool operations update pool level indices and position level fee bases. Pools do not take on credit exposure from direct agreements. Managed pools extend this model by introducing a manager address, mutable managed configuration, and a whitelist keyed to Position NFTs; membership-gated pools revert auto-join for non-whitelisted positions unless the manager disables gating. Unmanaged pools remain permissionless and immutable. Pools also support an atomic borrow and repay primitive (flash loan) that allows external contracts to borrow pool liquidity within a single transaction, subject to repayment plus fee before the transaction completes. Flash loans are constrained by the same reconciliation and encumbrance invariants as other pool operations: they do not create open ended debt, do not modify encumbrance variables, and must preserve trackedBalance_P consistency.

Direct agreements and the loan offer book. Direct is a bilateral agreement layer enabling cross asset credit without price oracles. It includes a permissionless offer mechanism that functions like a CLOB for loan terms, where *either side* can post executable offers. The same mechanism also supports yield-bearing limit orders for spot trading.

- **Lender posted offers (lend offers).** A lender posts an offer by escrowing principal from a pool balance associated with their position. Any borrower that meets the stated collateral requirements can accept, locking collateral and activating the agreement.
- **Borrower posted offers (borrow offers).** A borrower posts an offer to borrow by locking collateral up front. Any lender can accept by escrowing principal and activating the agreement.

Both offer types encode principal, collateral, interest, maturity, grace period, and any additional constraints required to form a valid agreement. At acceptance, principal and collateral become mutually encumbered under the agreement and are governed by the same deterministic settlement state machine. This symmetric offer model is a central mechanism for price discovery because users can place and take offers on either side, selecting rates and terms directly rather than being assigned rates by a reactive utilization function.

3.2.1 Yield-Bearing Limit Orders and Spot Trading

The protocol supports yield-bearing limit orders (YBLO) as a lightweight alternative to agreement-based offers. YBLOs enable spot trading while escrowed funds continue earning yield in pools:

- **Yield-bearing mechanism.** Escrowed funds remain in pool accounting and continue accruing fee index yield while encumbered, making the orders "yield-bearing" during their lifetime.

- **Two execution modes.** Offers can be filled either as (1) credit agreements with options-like outcomes through the full agreement state machine, or (2) spot trades via direct asset transfers without creating agreement objects.
- **Gas efficiency.** Spot execution bypasses agreement creation and state transitions, targeting approximately 200k gas for typical fills compared to 550k gas for agreement-based fills.
- **Agent compatibility.** The deterministic, executable nature makes YBLOs suitable for automated market making and algorithmic trading strategies.

This dual-mode execution enables the protocol to serve both credit markets (with time-based settlement) and spot markets (with immediate settlement) using the same underlying offer infrastructure.

Index module. EQUALINDEX is a tokenized basket module with its own fee pots and fee charging mechanisms. While logically separable, it shares design motifs with EQUALIS including deterministic fee accounting, explicit charging, and modular facet boundaries. In this paper we treat it as a companion system that can be used as an underlying asset in pools or as an asset held by positions, subject to token allowlisting and risk policies.

3.3 Positions as Transferable Account Containers

A Position NFT represents ownership of a protocol defined account container. Unlike an EOA based account model, a position is a portable state bundle. Ownership transfer transfers both assets and obligations that are scoped to that position. This design has two implications:

- **Composability.** Positions can be traded or integrated into higher level strategies without requiring protocol specific migration logic.
- **Local settlement.** Defaults and agreement settlement can be expressed as modifications to position state without requiring global bookkeeping or socialized loss mechanisms.

The protocol treats positions as the atomic unit of risk. Pools and agreements do not reason about identities beyond the position identifier.

3.4 Dual Index Accounting

Each pool maintains two indices.

Fee index. The fee index I_P^{fee} distributes system generated fees to positions. Fees accrue as a function of a position's fee base $B_{x,P}^{\text{fee}}$. A normalized fee base is used to prevent recursive amplification when a position increases its notional through internal movements.

Maintenance index. The maintenance index I_P^{mtn} applies proportional charging over time. Conceptually, it is an AUM like fee applied to relevant balance components. The separation between fee distribution and maintenance charging makes it possible to reason about protocol revenue and user yield without conflating charging with distribution.

Index updates are designed to be local and composable, and they occur at transaction time, not continuously. Later sections specify update rules and explain how these indices interact with deposits, withdrawals, borrows, and direct flows.

3.5 State Machines

3.5.1 Pooled Credit State

For pooled operations, the position's state in a pool is defined by deposited principal $\text{dep}_{x,P}$, pooled debt $\text{debt}_{x,P}$, and encumbrance variables such as $\text{locked}_{x,P}$ and $\text{offerEscrow}_{x,P}$. Deposits increase dep, withdrawals decrease dep subject to encumbrance constraints, borrows increase debt, and repayments decrease debt.

Defaults for pooled debt are handled through bounded settlement rules. The protocol does not liquidate assets via market auctions. Enforcement modifies the position's state according to predefined rules and does not socialize losses across depositors.

3.5.2 Direct Agreement State

Direct agreements are represented as explicit objects with the following high level states:

- **Offered:** an offer exists in the loan offer book. For lend offers, principal is escrowed. For borrow offers, collateral is locked.
- **Active:** an offer has been accepted, collateral is locked, principal has been transferred, and the agreement has a maturity time.
- **Repaid:** borrower repays principal before the recovery window and collateral is unlocked.
- **Exercised:** borrower opts to forfeit collateral early, terminating the agreement.
- **Recovered:** after maturity plus grace period, lender recovers locked collateral.

Only one terminal state is reachable for any agreement, and the protocol ensures that the allowed caller and time condition for each transition are explicit. The agreement's principals are reflected into per pool encumbrance variables, ensuring that pooled withdrawals cannot violate direct agreement constraints.

3.6 Core Invariants

We preview the key invariants that later sections formalize and use for proof sketches.

3.6.1 Pool Balance Reconciliation

For each pool P, internal accounting should reconcile to token balances up to explicitly defined reserves:

$$\text{trackedBalance}_P \approx \text{balanceOf}(\tau(P), P). \tag{1}$$

The precise reconciliation rule depends on transfer semantics and on any fee pots or reserves used by the pool. For supported assets, the protocol enforces reconciliation through controlled transfers and by updating trackedBalance_P in the same transaction as any token movement.

3.6.2 Encumbrance Constraint

For any position x and pool P, the withdrawable amount is bounded by deposited principal minus all encumbrances:

$$\text{withdrawable}_{x,P} \le \text{dep}_{x,P} - \Big(\text{locked}_{x,P} + \text{offerEscrow}_{x,P} + \text{directLent}_{x,P}\Big). \tag{2}$$

The protocol enforces that withdrawals and new obligations cannot reduce the position below this bound. The exact set of encumbrance terms may differ by operation, but the design principle is invariant: anything unavailable for withdrawal must be explicitly represented.

3.6.3 Pool Isolation

Encumbrances and debts are scoped by pool. Obligations created in pool P_i cannot be implicitly satisfied by assets in pool P_j. Cross pool relationships exist only through explicit direct agreements that lock collateral in the correct pool and update the correct encumbrance variables.

3.6.4 No Socialized Loss

Failures of a position to meet obligations must not directly reduce another position's deposited principal. In pooled lending, enforcement occurs through the defaulting position's state and through bounded settlement rules. In direct agreements, credit risk is bilateral between the two positions. The pooled system does not absorb direct agreement losses.

3.6.5 Deterministic Settlement

Agreement settlement depends only on:

- the agreement parameters selected at origination,
- the current block timestamp relative to maturity and grace period,
- explicit calls to the state machine transitions.

The protocol does not use reactive triggers such as price thresholds or liquidation auctions to enforce settlement.

"'latex

4 Equalis Pooled Credit Primitive

This section specifies the pooled credit layer of EQUALIS. Pools are single asset, positions are the unit of accounting, and safety properties reduce to explicit constraints over per position balances, per position encumbrances, and pool level reconciliation. The pooled layer supports two credit products: fixed term loans and self secured revolving credit lines with periodic payments.

4.1 Pool Definition and Scope

A pool P is parameterized by an underlying ERC20 like token $\tau(P)$ and configuration values that bound or price pooled credit. Pools do not depend on external prices for enforcement. In particular, pooled borrowing does not use collateral liquidation into the market to restore solvency. Instead, pooled borrowing is constrained by deterministic accounting, explicit encumbrance tracking, and bounded default settlement rules.

Pool state includes:

- trackedBalance_P, an internal accounting variable intended to match the pool's actual token balance after every operation that transfers $\tau(P)$.
- I_P^{fee} and I_P^{mtn}, two indices used to distribute fees and to apply maintenance charging.
- pool configuration parameters, including those that govern maintenance charging, fee collection, payment schedule rules for revolving credit, and default rules.

4.2 Position State in a Pool

For a given position x and pool P, the pooled credit state is represented by the following variables:

- $\mathrm{dep}_{x,P}$, deposited principal credited to x in pool P.
- $\mathrm{debt}_{x,P}$, outstanding pooled debt owed by x to pool P.
- $\mathrm{locked}_{x,P}$, principal locked as collateral for obligations that restrict withdrawal.
- $\mathrm{offerEscrow}_{x,P}$, principal escrowed to fund posted direct offers.
- $\mathrm{directLent}_{x,P}$ and $\mathrm{directBorrowed}_{x,P}$, principal amounts associated with active direct agreements.
- $B^{\mathrm{fee}}_{x,P}$, the normalized fee base used for fee index accrual.
- snapshots of indices needed to settle accrued yield and accrued maintenance charging.

For revolving credit lines, the position additionally tracks payment schedule state:

- $\mathrm{nextDue}_{x,P}$, timestamp of the next required periodic payment.
- $\mathrm{arrears}_{x,P}$, accumulated unpaid periodic amounts, if any.
- $\mathrm{apr}_{x,P}$ and product parameters needed to compute periodic dues, as defined by the pool and the credit line terms.

The set of encumbrance variables is intentionally explicit. Any quantity that cannot be withdrawn must appear as a first class state variable and must be enforced as a hard constraint during state transitions.

4.3 Core Operations

This subsection describes the pooled operations at the level of state transitions. Each operation is intended to be constant time in the number of positions and agreements.

4.3.1 Deposit

A deposit transfers $\tau(P)$ into the pool and credits the position:

$$\mathrm{dep}_{x,P} \leftarrow \mathrm{dep}_{x,P} + a \tag{3}$$

and updates pool reconciliation:

$$\mathrm{trackedBalance}_P \leftarrow \mathrm{trackedBalance}_P + a. \tag{4}$$

Deposits may also update fee base and index snapshots for x to ensure fee and maintenance accounting remain consistent.

4.3.2 Withdraw

A withdrawal transfers $\tau(P)$ out of the pool and debits the position, subject to encumbrance constraints.

Define the withdrawable amount:

$$\text{withdrawable}_{x,P} = \text{dep}_{x,P} - \Big(\text{locked}_{x,P} + \text{offerEscrow}_{x,P} + \text{directLent}_{x,P}\Big). \tag{5}$$

A withdrawal of amount a is valid only if:

$$a \leq \text{withdrawable}_{x,P}. \tag{6}$$

If valid, the protocol applies:

$$\text{dep}_{x,P} \leftarrow \text{dep}_{x,P} - a, \qquad \text{trackedBalance}_P \leftarrow \text{trackedBalance}_P - a. \tag{7}$$

This constraint is the primary mechanism that prevents a position from withdrawing principal that is encumbered by pooled credit collateral locks, direct offers, or active direct agreements.

4.3.3 Credit Origination

Equalis pooled borrowing is available in two forms. Both are self secured in the sense that the collateral used to support the credit is locked principal within the same pool asset. This keeps enforcement local to pool accounting and avoids cross asset price dependencies.

Fixed term borrow. A fixed term borrow increases pooled debt and transfers principal out of the pool:

$$\text{debt}_{x,P} \leftarrow \text{debt}_{x,P} + a, \qquad \text{trackedBalance}_P \leftarrow \text{trackedBalance}_P - a. \tag{8}$$

At origination, the protocol locks the required collateral by increasing $\text{locked}_{x,P}$, which reduces $\text{withdrawable}_{x,P}$ as defined in Equation 5. Fixed term borrows are repaid according to their term rules, and failure to meet the term triggers bounded default settlement.

Revolving credit line. A revolving credit line is an open ended borrow facility with a deterministic periodic payment requirement. Borrowers can draw, repay, and redraw up to an allowed credit limit so long as collateral and payment constraints remain satisfied.

A draw on the line has the same balance effects as borrowing:

$$\text{debt}_{x,P} \leftarrow \text{debt}_{x,P} + a, \qquad \text{trackedBalance}_P \leftarrow \text{trackedBalance}_P - a. \tag{9}$$

The distinguishing feature is the payment schedule state. At origination, the line sets $\text{nextDue}_{x,P}$ and defines the periodic amount computation. Periodic payments are assessed on a fixed cadence, for example monthly, and are enforced by restricting further draws and by enabling bounded default handling when the borrower becomes delinquent.

Credit limit and self secured collateral. For both products, the available credit is determined by collateral locked in the same pool:

$$\text{locked}_{x,P} \leftarrow \text{locked}_{x,P} + c \tag{10}$$

where c is chosen to satisfy the pool's LTV and product constraints. Because $\text{locked}_{x,P}$ is part of Equation 5, collateral used to secure pooled credit is not withdrawable while the obligation exists. This is the core enforcement mechanism for self secured credit.

4.3.4 Repay

Repayment transfers $\tau(P)$ into the pool and decreases pooled debt:

$$\mathrm{debt}_{x,P} \leftarrow \mathrm{debt}_{x,P} - a, \qquad \mathrm{trackedBalance}_P \leftarrow \mathrm{trackedBalance}_P + a. \tag{11}$$

For fixed term borrows, the protocol may require repayment by a maturity condition. For revolving credit, repayment can occur at any time, but borrowers must also satisfy the periodic payment requirement described below.

4.3.5 Flash Loan

A flash loan is an atomic borrow and repay operation that completes within a single transaction. The pool lends amount a of $\tau(P)$ to a receiver contract, which must return $a + \mathrm{fee}$ before the transaction completes. Flash loans do not create open ended debt and do not interact with encumbrance variables.

Process. The flash loan operation proceeds as follows:

1. The pool verifies $a \leq \mathrm{trackedBalance}_P$ (sufficient liquidity) and that the configured fee rate $\mathrm{flashLoanFeeBps}_P > 0$.
2. The pool computes $\mathrm{fee} = a \cdot \mathrm{flashLoanFeeBps}_P/10000$.
3. The pool transfers a of $\tau(P)$ to the receiver contract.
4. The pool invokes the receiver's callback, allowing arbitrary execution.
5. The pool pulls $a + \mathrm{fee}$ from the receiver via explicit transfer.
6. The pool verifies the actual token balance satisfies the repayment requirement.
7. On success, the pool updates $\mathrm{trackedBalance}_P \leftarrow \mathrm{trackedBalance}_P + \mathrm{fee}$ and routes the fee to the treasury via the standard fee accrual mechanism.

If repayment is insufficient or the callback reverts, the entire transaction reverts and state is unchanged.

Invariants preserved. Flash loans preserve the same reconciliation invariant as other pool operations:

$$\mathrm{trackedBalance}_P^{\mathrm{after}} = \mathrm{trackedBalance}_P^{\mathrm{before}} + \mathrm{fee}. \tag{12}$$

Critically, flash loans do not modify any encumbrance variables ($\mathrm{locked}_{x,P}$, $\mathrm{offerEscrow}_{x,P}$, $\mathrm{directLent}_{x,P}$) and do not create position level debt ($\mathrm{debt}_{x,P}$). The borrowed amount exists only transiently within the transaction and is not represented in protocol accounting state.

Anti split protection. The protocol optionally enforces anti split protection via $\mathrm{flashLoanAntiSplit}_P$. When enabled, a receiver address can execute at most one flash loan per block per pool, preventing attackers from splitting a large loan into multiple smaller loans to reduce per loan fees or to manipulate state between loans within the same block.

Fee routing. Flash loan fees are routed to the protocol treasury via the standard fee accrual mechanism. Unlike other fee sources, flash loan fees do not impact the pool's fee index for position yield distribution; they accrue directly to protocol revenue.

4.3.6 Periodic Payments for Revolving Credit

A revolving credit line requires a periodic payment on a deterministic schedule, for example monthly. Conceptually, the periodic amount is the line's due amount for the period, derived from the line's parameters and the outstanding debt over the period. The protocol enforces this requirement via state transitions that:

- compute the amount due for the period,
- accept payment which increases trackedBalance_P and reduces $\text{arrears}_{x,P}$ and or other due tracking,
- advance $\text{nextDue}_{x,P}$ to the next period.

If a payment is missed, the protocol records delinquency by increasing $\text{arrears}_{x,P}$ and restricting actions that would increase risk, such as additional draws or collateral withdrawals that would reduce the secured buffer.

This design ensures that revolving credit is meaningfully distinct from open ended debt with no enforcement until some external price trigger. The enforcement trigger is the schedule itself.

4.4 Dual Index Accounting

Each pool maintains two indices to separate fee distribution from maintenance charging. Both indices update at transaction time and do not require periodic keeper execution.

4.4.1 Fee Index

The fee index I_P^{fee} represents cumulative distributed fees per unit of normalized fee base. Fees are generated by protocol defined actions, for example direct agreement premiums, system fees, or other configured sources. When fees of amount f are added to the pool's distributable fee pot, the index increases by:

$$\Delta I_P^{\text{fee}} = \frac{f}{\sum_x B_{x,P}^{\text{fee}}}, \qquad I_P^{\text{fee}} \leftarrow I_P^{\text{fee}} + \Delta I_P^{\text{fee}}. \tag{13}$$

A position x accrues fee yield proportional to its normalized fee base:

$$\text{feeAccrued}_{x,P} = B_{x,P}^{\text{fee}} \cdot \left(I_P^{\text{fee}} - I_{x,P,\text{snap}}^{\text{fee}}\right) \tag{14}$$

where $I_{x,P,\text{snap}}^{\text{fee}}$ is the last settled snapshot for x in pool P.

Normalized fee base. A critical design goal is to prevent recursive amplification of fee accrual through internal self dealing movements that increase notional without increasing net contributions. The protocol therefore uses a normalized fee base $B_{x,P}^{\text{fee}}$ that is not equal to raw deposited principal. The exact definition is specified in Section 4.5, but the constraint we rely on is:

$$B_{x,P}^{\text{fee}} \leq \text{dep}_{x,P} \tag{15}$$

and base updates are bounded such that looping behaviors cannot increase total fee base without bringing additional net principal into the system.

4.4.2 Maintenance Index

The maintenance index I_P^{mtn} represents cumulative maintenance charge per unit of a configured maintenance base. Maintenance charging is conceptually an AUM like fee. When maintenance fees of amount m are assessed, the index increases:

$$\Delta I_P^{\mathrm{mtn}} = \frac{m}{\sum_x B_{x,P}^{\mathrm{mtn}}}, \qquad I_P^{\mathrm{mtn}} \leftarrow I_P^{\mathrm{mtn}} + \Delta I_P^{\mathrm{mtn}} \tag{16}$$

where $B_{x,P}^{\mathrm{mtn}}$ denotes the configured maintenance base and is specified alongside the fee base in Section 4.5. A position settles maintenance charges by applying:

$$\mathrm{mtnOwed}_{x,P} = B_{x,P}^{\mathrm{mtn}} \cdot \left(I_P^{\mathrm{mtn}} - I_{x,P,\mathrm{snap}}^{\mathrm{mtn}}\right). \tag{17}$$

4.4.3 Yield Settlement and Principal Conversion

The protocol supports explicit settlement actions that realize accrued fee yield and update position balances. One canonical operation is to roll accrued yield into deposited principal:

$$\mathrm{dep}_{x,P} \leftarrow \mathrm{dep}_{x,P} + \mathrm{feeAccrued}_{x,P} \tag{18}$$

with corresponding updates to snapshots and fee bases. This action is optional and deterministic. Users may also withdraw accrued yield as part of a withdrawal flow, subject to the same encumbrance constraints.

4.5 Fee Base and Maintenance Base Definition

This subsection defines the precise bases used for the two indices, including how locked principal, escrowed principal, and direct agreement principals contribute or do not contribute. The intent is to ensure:

- fee accrual corresponds to protocol configured notions of participation,
- maintenance charging corresponds to protocol configured notions of capital usage,
- base updates are local and do not require iteration over global position sets,
- base normalization prevents recursive amplification from looping behaviors.

4.6 Default and Delinquency Handling

Pooled credit enforcement is expressed through deterministic rules over time and state. This includes both fixed term default and revolving credit delinquency.

Fixed term default. If a fixed term obligation reaches its maturity conditions without repayment, the protocol applies bounded, rules based settlement rather than liquidation auctions. The penalty is computed from a basis defined at origination, such as principalAtOpen, and the rate is fixed by configuration and does not depend on external prices.

Revolving credit delinquency. If a revolving credit line misses a required periodic payment, the position enters a delinquent state represented by $\mathrm{arrears}_{x,P} > 0$ and an overdue $\mathrm{nextDue}_{x,P}$. While delinquent, the protocol restricts risk increasing actions, such as new draws and withdrawals that would reduce the secured buffer. If delinquency persists beyond a configured tolerance window, the protocol enables bounded enforcement analogous to default settlement, again without liquidation auctions and without relying on external market prices.

Respecting direct encumbrances. All enforcement must respect assets encumbered by direct agreements and posted offers. Principal represented by $\text{locked}_{x,P}$ and $\text{offerEscrow}_{x,P}$ is not freely seizable if doing so would violate an active direct agreement's collateral lock or offer escrow constraint. This requirement is enforced by the same encumbrance accounting used for withdrawals. Enforcement modifies only the position's unencumbered state and must preserve invariants required for active direct agreements.

Penalty distribution. When enforcement applies, the seized remainder and the penalty are removed from the borrower's principal. The penalty component is split deterministically: 10% to the enforcing caller (incentive), 63% to the pool FeeIndex, 9% to the protocol treasury, and 18% to the Active Credit Index. FeeIndex and Active Credit portions remain inside protocol accounting; treasury and enforcer shares transfer out.

4.7 Correctness Preview

We preview the core correctness properties of the pooled credit primitive. Full statements and proof sketches appear in Section 7.

- **Reconciliation.** trackedBalance_P is updated in the same transaction as any transfer of $\tau(P)$, preserving balance alignment for supported tokens.
- **Encumbrance safety.** No valid transition allows $\text{dep}_{x,P}$ to fall below total encumbrances. Withdrawals, new draws, and enforcement are constrained by Equation 5.
- **Schedule enforcement.** Revolving credit lines have explicit periodic payment state, and delinquency is represented and enforced deterministically.
- **Pool isolation.** All variables are indexed by pool, so obligations and encumbrances in pool P_i cannot be satisfied or violated using assets in pool P_j.
- **No socialized loss.** Enforcement modifies only the defaulting or delinquent position's state under bounded rules and does not impose implicit losses on passive depositors.

Section 5 specifies the direct agreement layer that interacts with pooled state through $\text{locked}_{x,P}$, $\text{offerEscrow}_{x,P}$, and the direct principal variables.

5 Equalis Direct: Oracle Free Bilateral Credit Agreements

This section specifies Direct, the bilateral agreement layer that enables cross asset credit without price oracles. Direct is designed as an executable loan offer book where either side can post offers, and where settlement is deterministic and time based rather than liquidation based. A distinguishing property is that collateral locked by the borrower remains in the pool accounting system and continues to accrue pool level yield during the lifetime of the agreement, subject to the pool's fee base and maintenance base definitions.

5.1 Design Goals

Direct is built to satisfy the following goals:

- **Oracle free cross asset credit.** Agreement correctness and enforcement must not depend on external prices.

- **Bilateral risk isolation.** The pooled system must not underwrite the credit risk of direct agreements. Risk is isolated to the two positions that enter the agreement.
- **Executable offer book.** Either side can post offers that are executable by counterparties, enabling price discovery for loan terms without utilization curves.
- **Yield-bearing limit orders and spot trading.** Offers can represent either credit agreements or spot exchange, with escrowed funds earning yield while waiting for execution.
- **Agent-friendly execution.** The protocol supports standing executable offers suitable for automated market makers and algorithmic trading strategies.
- **Deterministic settlement.** Repayment, early exercise, and recovery are governed by a state machine defined by parameters at origination and triggered by time and explicit calls.
- **Explicit encumbrance accounting.** Principal and collateral used by direct agreements must be represented as encumbrances in their respective pools, preventing withdrawal or reuse while obligated.
- **Collateral continues earning.** Borrower collateral remains part of pool accounting and can continue to accrue pool yield, allowing a borrower to earn on locked collateral while deploying the borrowed funds.

5.2 Objects and Parameters

A direct agreement is created by accepting an offer. The system defines two offer types that are symmetric in intent but differ in which side escrows at posting time. The resulting agreements have options-like characteristics: borrowers can choose between repayment (preserving collateral) and exercise (forfeiting collateral), similar to early exercise of a secured option.

5.2.1 Offer Types

Lend offer. A lender posted offer escrows principal p from a lender position L in a principal pool P_p such that $\tau(P_p) = \tau_p$. The offer specifies required collateral c from a borrower position B in a collateral pool P_c such that $\tau(P_c) = \tau_c$, where τ_c may differ from τ_p.

Borrow offer. A borrower posted offer locks collateral c from a borrower position B in a collateral pool P_c. The offer specifies desired principal p to be escrowed by any accepting lender position L in a principal pool P_p.

Both offer types result in the same agreement object once accepted.

Tranche and ratio tranche offers. Beyond single fill offers, lenders may post tranche offers that reserve a principal tranche for multiple fills. In the ratio tranche variant, the lender posts a price ratio (n, d) representing collateral per unit principal, a principal cap, and a minimum principal-per-fill. Borrowers can draw any amount up to the cap; required collateral is computed as collateral = principal $\times\, n/d$ at acceptance. Escrowed principal is reserved at post time and decremented per fill; acceptance re-encumbers the filled slice into active direct principal.

Limit orders. In addition to agreement-based offers, the protocol supports limit orders: lightweight trading orders that encumber capital at a specified price ratio without creating agreements. Fills resolve via direct asset transfers rather than agreement creation, providing a gas-efficient trading path. A maker posts an order specifying a price ratio, cap amount, and minimum fill size. Takers can fill any amount within $[\text{minFill}, \text{remaining}]$. The counterparty amount is computed from the price ratio, a flat basis points fee is deducted from the taker's received amount, and assets transfer directly between positions. Encumbered capital reuses existing mappings (offerEscrow for lender-side, locked for borrower-side) and continues earning fee index yield while waiting for fills. When a position NFT is transferred, all outstanding limit orders are automatically cancelled and encumbrance released.

Yield-Bearing Limit Orders (YBLO). The user-facing form of limit orders is termed yield-bearing limit orders (YBLO). The "yield-bearing" property arises mechanically: escrowed funds remain in pool accounting and continue accruing fee index yield while encumbered as offers. This distinguishes YBLOs from traditional limit orders where capital sits idle. YBLOs behave like buy/sell walls when interpreted as exchange offers, enabling users to:

- Post standing buy or sell orders at specified price ratios
- Earn pool yield on escrowed capital while waiting for fills
- Execute spot trades without agreement state machine overhead
- Support partial fills with automatic price calculation

The same offer infrastructure supports both credit agreements (with full state machine) and spot trading (direct transfers), providing flexibility for different use cases.

5.2.2 Agreement Parameters

Each agreement A is parameterized by:

- principal amount p in asset τ_p and pool P_p,
- collateral amount c in asset τ_c and pool P_c,
- premium or interest amount π in asset τ_p (realized at acceptance),
- maturity timestamp t_m,
- grace period length Δ_g,
- optional lender call permission allowCall (whether the lender may accelerate maturity to current block time before t_m),
- identifiers of lender position L and borrower position B,
- any constraints required to validate acceptance, including pool allowlists and per position caps.

Premium semantics. The protocol treats the interest component as a premium realized at agreement activation. This simplifies enforcement because repayment is defined over principal only. The borrower pays π at acceptance, and it is not refunded upon early repayment.

5.3 Yield on Locked Collateral

Unlike liquidation based lending systems where collateral is functionally inert while posted, Equalis preserves the borrower's pool participation on locked collateral. When a borrower locks collateral c in pool P_c, that collateral remains recorded in dep_{B,P_c} and is merely made non withdrawable by increasing locked_{B,P_c}. Because fee accrual is computed from pool level indices applied to a position's bases, locked collateral can continue to accrue yield for the borrower for the lifetime of the agreement, subject to how the pool defines the fee base and maintenance base.

The key mechanical point is that locking collateral is an *encumbrance* operation, not a transfer to an external vault. Therefore, the collateral remains inside the accounting domain that accrues pool yield:

$$\mathrm{locked}_{B,P_c} \leftarrow \mathrm{locked}_{B,P_c} + c, \qquad \mathrm{dep}_{B,P_c} \text{ unchanged.} \tag{19}$$

This property matters because it changes the borrower's economic tradeoff. A borrower can enter a single direct agreement, keep their collateral earning in the pool, and deploy the borrowed principal for any external use. The total borrower outcome becomes:

- gains or losses on how the borrowed funds are deployed off protocol,
- plus any pool yield accrued on the locked collateral during the agreement lifetime,
- minus the premium π paid at acceptance and any maintenance charges assessed by the collateral pool.

Whether locked collateral contributes fully, partially, or not at all to fee and maintenance bases is a pool level policy choice. This paper states the mechanism as a design goal and defers the exact base definitions to Section 4.5. The important invariant is that the accounting representation allows continued accrual without violating withdrawal safety, because encumbrance reduces withdrawability even if the balance continues to earn.

5.4 State Machine

Each offer and agreement is governed by a state machine. We distinguish the offer lifecycle from the agreement lifecycle.

5.4.1 Offer Lifecycle

An offer O exists in one of the following states:

- **Open:** posted and available to be accepted.
- **Cancelled:** removed by the posting party prior to acceptance.
- **Filled:** accepted and transformed into an active agreement.

Encumbrances during Open. For a lend offer, the lender's principal is escrowed and reflected as an encumbrance:

$$\mathrm{offerEscrow}_{L,P_p} \leftarrow \mathrm{offerEscrow}_{L,P_p} + p. \tag{20}$$

For a borrow offer, the borrower's collateral is locked and reflected as an encumbrance:

$$\mathrm{locked}_{B,P_c} \leftarrow \mathrm{locked}_{B,P_c} + c. \tag{21}$$

Cancellation reverses these encumbrances.

5.4.2 Agreement Lifecycle

Once an offer is filled, an agreement A enters the **Active** state. It can then transition to exactly one terminal state.

- **Active:** agreement is live, collateral is locked, principal has been transferred.
- **Repaid:** borrower repaid principal before lender recovery is permitted.
- **Exercised:** borrower voluntarily forfeited collateral early to terminate the agreement.
- **Recovered:** lender recovered collateral after maturity plus grace period.

Define the recovery time:

$$t_r = t_m + \Delta_g. \tag{22}$$

The allowed transitions are:

- **Active** $\rightarrow$ **Repaid**, callable by borrower position owner, valid only when $t < t_r$ and principal is transferred to lender; if allowEarlyRepay is false, repayment is only valid from $t \in [t_m - 24\text{h}, t_r)$.
- **Active** $\rightarrow$ **Exercised**, callable by borrower position owner; before t_m this requires allowEarlyExercise = true, and during the grace window $t \in [t_m, t_r)$ it is permitted regardless of allowEarlyExercise.
- **Active** $\rightarrow$ **Recovered**, callable by lender position owner, valid only when $t \geq t_r$.
- **Active** (optional lender call) updates $t_m \leftarrow$ now if allowCall = true and $t < t_m$, callable by lender position owner.

This structure ensures determinism at boundaries. Before recovery time, the borrower controls termination through repayment or exercise. After recovery time, the lender controls termination through recovery.

5.5 Core Flows

This subsection describes the core flows and the encumbrance updates they induce. We use L for the lender position, B for the borrower position, P_p for the principal pool, and P_c for the collateral pool.

5.5.1 Post Lend Offer

To post a lend offer, L escrows principal p in P_p:

$$\text{offerEscrow}_{L,P_p} \leftarrow \text{offerEscrow}_{L,P_p} + p. \tag{23}$$

The offer is recorded as **Open**. No borrower state changes occur at posting.

5.5.2 Post Borrow Offer

To post a borrow offer, B locks collateral c in P_c:

$$\text{locked}_{B,P_c} \leftarrow \text{locked}_{B,P_c} + c. \tag{24}$$

The offer is recorded as **Open**. No lender state changes occur at posting.

5.5.3 Cancel Offer

The posting party can cancel an **Open** offer. Cancellation reverses the posting encumbrance.

For lend offers:

$$\text{offerEscrow}_{L,P_p} \leftarrow \text{offerEscrow}_{L,P_p} - p. \tag{25}$$

For borrow offers:

$$\text{locked}_{B,P_c} \leftarrow \text{locked}_{B,P_c} - c. \tag{26}$$

5.5.4 Accept Lend Offer

A borrower accepts a lend offer by providing collateral and paying premium.

At acceptance:

1. Borrower locks collateral in P_c:

$$\text{locked}_{B,P_c} \leftarrow \text{locked}_{B,P_c} + c. \tag{27}$$

2. Lender offer escrow is converted into an active lent principal amount:

$$\text{offerEscrow}_{L,P_p} \leftarrow \text{offerEscrow}_{L,P_p} - p, \qquad \text{directLent}_{L,P_p} \leftarrow \text{directLent}_{L,P_p} + p. \tag{28}$$

   Lender principal and fee base reduce immediately by p; the pool's Active Credit base and the lender's P2P Active Credit state increase by p with weighted dilution.

3. Borrower direct borrowed amount increases:

$$\text{directBorrowed}_{B,P_p} \leftarrow \text{directBorrowed}_{B,P_p} + p. \tag{29}$$

   If $\tau_p = \tau_c$, the borrower's same-asset Active Credit base increases by p with the same time gate and dilution; cross-asset loans do not add to borrower Active Credit.

4. Principal p is transferred to the borrower, and premium π is transferred from the borrower to the lender or to the pool's fee mechanism as configured.

The agreement enters **Active** with maturity t_m and grace Δ_g.

5.5.5 Accept Borrow Offer

A lender accepts a borrow offer by escrowing principal and paying out principal. The borrower's collateral is already locked by posting.

At acceptance:

1. Lender escrows principal and immediately converts it to active lent principal:

$$\text{directLent}_{L,P_p} \leftarrow \text{directLent}_{L,P_p} + p. \tag{30}$$

   Lender principal and fee base reduce immediately by p; the pool's Active Credit base and the lender's P2P Active Credit state increase by p with weighted dilution.

2. Borrower direct borrowed amount increases:

$$\text{directBorrowed}_{B,P_p} \leftarrow \text{directBorrowed}_{B,P_p} + p. \tag{31}$$

   If $\tau_p = \tau_c$, the borrower's same-asset Active Credit base increases by p; cross-asset loans do not add to borrower Active Credit.

3. Principal p is transferred to the borrower, and premium π is transferred from the borrower to the lender or fee mechanism as configured.

The agreement enters **Active**. Because collateral was already locked during posting, the borrower experiences a two phase process: collateral lock first, then principal receipt upon lender acceptance.

5.5.6 Repay

To repay an **Active** agreement before recovery time, the borrower transfers principal p in τ_p to the lender and the protocol releases collateral.

State updates include:

$$\text{directBorrowed}_{B,P_p} \leftarrow \text{directBorrowed}_{B,P_p} - p, \tag{32}$$

$$\text{directLent}_{L,P_p} \leftarrow \text{directLent}_{L,P_p} - p, \tag{33}$$

$$\text{locked}_{B,P_c} \leftarrow \text{locked}_{B,P_c} - c. \tag{34}$$

Lender P2P Active Credit base and state reduce by p; if $\tau_p = \tau_c$, borrower same-asset Active Credit base and state reduce by p.

Premium π is not refunded. Collateral yield accrued prior to repayment remains with the borrower position according to the pool's index accounting.

5.5.7 Exercise Early

Before recovery time, the borrower can choose to terminate the agreement by forfeiting collateral. This is economically similar to early exercise of a secured option.

State updates include:

$$\text{directBorrowed}_{B,P_p} \leftarrow \text{directBorrowed}_{B,P_p} - p, \tag{35}$$

$$\text{directLent}_{L,P_p} \leftarrow \text{directLent}_{L,P_p} - p, \tag{36}$$

$$\text{locked}_{B,P_c} \leftarrow \text{locked}_{B,P_c} - c. \tag{37}$$

Lender P2P Active Credit base and state reduce by p; if $\tau_p = \tau_c$, borrower same-asset Active Credit base and state reduce by p.

and the protocol transfers collateral c in τ_c to the lender. Any yield that accrued on the collateral while it remained in the pool accounting domain is already reflected in the borrower's position state prior to exercise, subject to the base definitions and settlement mechanics.

5.5.8 Lender Call (Optional)

If configured at origination with allowCall = true, the lender can accelerate maturity before t_m by calling the agreement. This updates the agreement's maturity to the current block time, leaving the grace window unchanged:

$$t_m \leftarrow \text{now}, \qquad t_r \leftarrow t_m + \Delta_g. \tag{38}$$

No principals move during a call. Encumbrances remain unchanged:

$$\text{directBorrowed}_{B,P_p} \text{ unchanged}, \tag{39}$$

$$\text{directLent}_{L,P_p} \text{ unchanged}, \tag{40}$$

$$\text{locked}_{B,P_c} \text{ unchanged}. \tag{41}$$

Economic effect: the borrower's repayment and exercise windows shift earlier, compressing optionality. Before the call, the borrower controls termination until t_r as usual; after the call, the new t_m and t_r bounds apply. Lender calls are invalid if $\text{allowCall} = \text{false}$ or if $t \geq t_m$ (maturity already reached).

5.5.9 Recover After Grace Period

After recovery time t_r, the lender can recover collateral.

State updates mirror exercise in terms of encumbrance removal:

$$\text{directBorrowed}_{B,P_p} \leftarrow \text{directBorrowed}_{B,P_p} - p, \tag{42}$$

$$\text{directLent}_{L,P_p} \leftarrow \text{directLent}_{L,P_p} - p, \tag{43}$$

$$\text{locked}_{B,P_c} \leftarrow \text{locked}_{B,P_c} - c. \tag{44}$$

Lender P2P Active Credit base and state reduce by p; if $\tau_p = \tau_c$, borrower same-asset Active Credit base and state reduce by p.

and collateral c is transferred to the lender. Collateral yield accrued before recovery remains accounted for according to the pool indices and the position's bases until the collateral leaves the position at recovery.

5.6 Interaction With Pooled State

Direct agreements interact with pooled state only through explicit encumbrance variables and principal movement constraints.

Withdrawal safety. Because collateral locks and offer escrows increase $\text{locked}_{x,P}$ and $\text{offerEscrow}_{x,P}$, Equation 5 prevents withdrawal of funds committed to direct agreements or offers.

Pool solvency isolation. The pool does not assume credit risk of a direct agreement. Principal lent via Direct is escrowed from a position's pool balance and is not treated as a pool wide obligation. If the borrower defaults in the direct sense, the lender receives collateral via recovery rather than a pool absorbing a loss.

5.7 Correctness Properties Preview

We preview the key properties that the direct layer must satisfy.

- **Single terminal state.** Each agreement reaches exactly one terminal state and cannot be settled twice.
- **Time gated permissions.** Before recovery time, only the borrower can terminate (subject to the early repay window if disabled and to the early exercise flag before maturity). During the grace window, borrower may repay or exercise; after recovery time, only the lender can terminate. An optional lender call may accelerate maturity if configured.
- **Encumbrance preservation.** While an offer is open or an agreement is active, the relevant principal and collateral remain encumbered and cannot be withdrawn or reused.
- **Collateral yield compatibility.** Collateral locking is implemented as an encumbrance within pooled accounting, enabling continued index based accrual while preserving withdrawal safety.

- **Bilateral risk isolation.** Direct credit risk is isolated to the two positions, and pool state remains solvent under agreement settlement.

Section 7 provides formal statements and proof sketches for these properties and relates them to the pooled invariants introduced in Section 4.

6 EqualIndex Module

This section describes EqualIndex, a tokenized basket module that is architecturally separable from Equalis but intentionally compatible with the same accounting philosophy: deterministic fee charging, explicit fee pots, and modular boundaries. We include it for two reasons. First, it is part of the broader system described by the project knowledge base. Second, it provides a concrete example of how deterministic fee distribution can be implemented without relying on reactive market triggers.

6.1 Motivation and Scope

Index products onchain frequently conflate three concerns: custody of underlying assets, fee extraction, and fee distribution to stakeholders. EqualIndex separates these concerns by maintaining explicit accounting for vault balances and fee pot balances. This makes it possible to reason about net asset value (NAV), fees owed, and fees distributable as distinct quantities rather than as implicit balance deltas.

The module supports:

- mint and burn of index shares against a defined basket,
- management fees and optional protocol fees,
- deterministic fee pot accounting for distribution,
- flash loans as an explicit feature with defined constraints.

6.2 High Level Architecture

EqualIndex is implemented as a modular system with clear responsibility boundaries. While the specific implementation can use a facet based pattern, the conceptual decomposition is:

- **Basket definition and validation:** asset list, weights, and constraints for mint and burn.
- **Vault accounting:** custody balances for underlying assets that back the index token supply.
- **Fee accounting:** management fee accrual and optional protocol fee extraction.
- **Fee pot distribution:** explicit balances reserved for distribution separate from vault balances.
- **Flash loan interface:** controlled temporary borrowing of underlying assets with a fee.

The key design intent is that fees do not silently erode vault custody balances without being represented as a distinct, distributable quantity.

6.3 Dual Balance Model: Vault vs Fee Pots

Let V_i denote the vault custody balance of asset τ_i held to back index shares, and let F_i denote the fee pot balance of asset τ_i accumulated for distribution.

The dual balance model enforces:

- **NAV backing:** index share value is derived from V_i holdings, not from F_i.
- **Fee clarity:** fees are accumulated into F_i explicitly, rather than being implied by a decrease in V_i.
- **Distribution separation:** distribution claims can be computed and settled against F_i without interfering with vault custody invariants.

This separation is aligned with the EQUALIS design choice that encumbered balances remain in the accounting domain but are tracked explicitly. In EQUALINDEX, fees remain in the contract domain but are tracked as fee pot balances rather than vault balances.

6.4 Mint and Burn Interfaces

Mint. Minting index shares requires depositing the appropriate basket amounts into the vault. The mint operation:

- validates deposit amounts against the basket definition,
- increases vault balances V_i by deposited amounts,
- mints index shares to the receiver,
- updates fee accounting state if fees are assessed at mint time.

Burn. Burning index shares redeems underlying assets from the vault. The burn operation:

- burns index shares from the holder,
- computes redeemable amounts according to basket rules,
- decreases vault balances V_i by redeemed amounts,
- transfers redeemed assets to the receiver.

If management fees accrue continuously by index accounting, burn operations may include settlement of fees up to the current time, moving accrued amounts into fee pot balances F_i.

6.5 Fee Accrual and Distribution

Management fees. Management fees can be expressed as a deterministic accrual rule over time and a defined base. Accrued fees are moved into fee pots:

$$F_i \leftarrow F_i + \Delta f_i, \qquad V_i \leftarrow V_i - \Delta f_i, \tag{45}$$

subject to constraints that maintain non negativity and preserve vault backing for outstanding shares.

Protocol fees. If enabled, a portion of accrued fees can be routed to a protocol fee pot or treasury. This is a configuration choice and should be represented explicitly in accounting rather than being implicit in vault balances.

Distribution mechanism. Distribution can be implemented as a claimable process where eligible holders redeem against F_i according to their entitlement. The important requirement is that distribution does not interfere with vault custody invariants, because it only consumes fee pot balances.

6.6 Flash Loans

The module supports flash loans as an explicit interface. A flash loan allows temporary borrowing of underlying assets within one transaction subject to repayment plus fee. A typical flash loan correctness condition is:

$$V_i^{\text{after}} \geq V_i^{\text{before}} + \text{fee}_i \tag{46}$$

for the borrowed asset τ_i, where the fee is credited to the appropriate pot, typically F_i.

Flash loans introduce additional reentrancy and callback risk surface. As a result, the module must enforce strict checks on repayment, and it should apply standard defenses such as non reentrancy guards, checks effects interactions patterns, and controlled external call boundaries.

Relationship to Equalis pool flash loans. Both Equalis pools and EqualIndex vaults support flash loans with similar constraints. In Equalis pools, flash loans operate on the single pool asset $\tau(P)$ and fees route to the protocol treasury. In EqualIndex, flash loans operate on the proportional basket of underlying assets and fees route to the fee pot F_i. Both implementations enforce diamond level reentrancy guards, explicit repayment verification, and atomicity guarantees. The key difference is that EqualIndex flash loans borrow a proportional bundle of assets rather than a single asset amount.

6.7 Security and Testing Posture

EqualIndex is designed to be evaluated as a deployed artifact. The security posture is structured around:

- **Access control correctness:** only authorized configuration changes, no privileged minting or draining paths.
- **Invariant preservation:** vault balances cannot go negative, fee pot balances cannot exceed what has been accrued, total share supply maps correctly to vault backing.
- **Token behavior constraints:** allowlisting policies for supported tokens to avoid fee on transfer and rebasing hazards unless explicitly handled.
- **Property based testing:** mint, burn, fee accrual, and flash loan properties are stated as invariants and validated across randomized sequences of operations.

This testing posture mirrors the Equalis approach where invariants are treated as first class specifications rather than as informal assumptions.

6.8 Integration Surface With Equalis

EqualIndex interacts with Equalis only through shared asset space and shared position ownership semantics.

- **Index shares as pool assets:** an index token can be the underlying asset $\tau(P)$ for a pool, subject to token allowlisting and transfer behavior compatibility.
- **Index shares as collateral:** positions may lock index shares as collateral within Direct agreements if a pool exists for that asset and the pool's policy allows it.
- **Fee accounting compatibility:** the explicit fee pot model aligns with Equalis's preference for explicit encumbrance and explicit distribution, enabling clearer reasoning about yield attribution when index shares are held by positions.

The remainder of this paper focuses on Equalis correctness. EqualIndex is included to demonstrate a consistent accounting philosophy across modules and to define the composability boundary conditions.

7 Formal Properties and Proof Sketches

This section states the key correctness properties of Equalis and provides proof sketches. The statements are written as protocol level claims over state transitions. Proof sketches are informal but structured to make assumptions explicit and to clarify which invariants must be preserved by implementation.

7.1 Preliminaries

Let S denote global protocol state and let $\to$ denote a valid state transition induced by a single successful transaction. For a transition $S \to S'$, we assume:

- all arithmetic is exact over integer units with explicit rounding rules,
- all external token transfers revert on failure for supported tokens,
- any required non reentrancy conditions hold for the transition.

We use P for a pool, $\tau(P)$ for its token, and x for a position. Encumbrance variables are as defined in previous sections.

7.2 Property 1: Pool Balance Reconciliation

Claim. For any supported token $\tau(P)$ and any transition that transfers $\tau(P)$ into or out of pool P, the pool's tracked balance remains consistent with the actual token balance:

$$\text{trackedBalance}_P(S') = \text{trackedBalance}_P(S) + \Delta_{\text{in}} - \Delta_{\text{out}} \tag{47}$$

and

$$\text{trackedBalance}_P(S') = \text{balanceOf}(\tau(P), P, S') \tag{48}$$

where Δ_{in} and Δ_{out} are the net token amounts transferred in and out during the transaction.

Proof sketch. Each protocol function that transfers tokens updates trackedBalance_P in the same control flow path as the transfer. For supported tokens, transfer execution is atomic with respect to transaction success: either the transfer and state update both occur, or the transaction reverts leaving both unchanged. Therefore, after the transfer, trackedBalance_P matches the actual balance. This property fails for fee on transfer and rebasing tokens, which motivates allowlisting and adapters.

7.3 Property 1a: Flash Loan Atomicity and Reconciliation

Claim. If a flash loan of amount a succeeds, then trackedBalance_P after the call equals trackedBalance_P before plus the fee. If repayment is insufficient, the transaction reverts and state is unchanged. Flash loans do not create position level debt or modify encumbrance variables.

$$\text{trackedBalance}_P^{\text{after}} = \text{trackedBalance}_P^{\text{before}} + \text{fee} \tag{49}$$

where $\text{fee} = a \cdot \text{flashLoanFeeBps}_P/10000$.

Proof sketch. The flash loan function is guarded by a diamond level non reentrancy modifier. The function transfers a to the receiver, invokes the callback, then explicitly pulls $a + \text{fee}$ from the receiver. Before returning, the function verifies that the actual token balance satisfies the repayment requirement. Only after this verification does the function update trackedBalance_P by adding the fee. If the balance check fails, the transaction reverts and no state changes persist. Because the function does not modify any position state variables ($\text{dep}_{x,P}$, $\text{debt}_{x,P}$, $\text{locked}_{x,P}$, etc.), flash loans are invisible to encumbrance accounting. The borrowed amount exists only transiently within the transaction scope. This makes flash loans a special case of Property 1 where the net effect is token in (fee) with no token out at transaction end.

7.4 Property 2: Encumbrance Safety and Withdrawal Correctness

Claim. For any position x and pool P, no valid transition allows the position to withdraw or otherwise reduce its pool credited balance below its total encumbrances. In particular, for any withdrawal of amount a:

$$a \leq \text{dep}_{x,P} - \Big(\text{locked}_{x,P} + \text{offerEscrow}_{x,P} + \text{directLent}_{x,P}\Big). \tag{50}$$

Proof sketch. Encumbrances are increased at the moment a constraint is created, namely when collateral is locked, when an offer is posted, or when a direct agreement becomes active. The withdrawal function checks Equation 5 against the requested amount a. Because the function is constant time and uses explicit encumbrance variables, no withdrawal path can ignore an active encumbrance. If a path attempted to reduce $\text{dep}_{x,P}$ without reducing encumbrances, it would violate the checked inequality and revert. Therefore, under the assumption that all state updates route through these guarded transitions, the property holds.

7.5 Property 3: Pool Isolation

Claim. Obligations and encumbrances are scoped by pool. A valid transition cannot cause assets in pool P_i to be withdrawn, seized, or released to satisfy an obligation created in pool P_j unless an explicit agreement locks assets in P_i.

Proof sketch. All balance and encumbrance variables are indexed by pool. Operations in pool P_i only read and write variables with index P_i, except when an operation explicitly spans pools, such as direct agreement acceptance which simultaneously updates P_p and P_c variables. Even in that case, the spanning behavior is explicit and updates both pools' encumbrances. There is no implicit cross pool netting or global health factor. Therefore, absent a spanning operation that references both pools, pool P_i state transitions are independent of P_j.

7.6 Property 4: No Socialized Loss

Claim. A position's failure to meet obligations does not directly reduce another position's deposited principal, except through protocol fees and explicitly agreed transfers.

Proof sketch. In pooled lending, default handling modifies only the defaulting position's state and does not create a shared bad debt pool that is repaid by depositors. In Direct, credit risk is bilateral and is resolved through collateral transfer between two positions. There is no mechanism by which losses in a direct agreement are charged to pool depositors. Therefore, any balance reduction for a non defaulting position must arise from fees or explicit transfers, not from third party default.

7.7 Property 5: Deterministic Direct Settlement

Claim. Each direct agreement reaches exactly one terminal state, and settlement depends only on agreement parameters, time relative to maturity and grace period, and explicit calls by authorized parties.

Proof sketch. The agreement state machine defines disjoint time gated transition sets. Before recovery time t_r, the borrower can terminate by repayment or exercise. After t_r, only the lender can terminate via recovery. The implementation enforces that terminal states are absorbing and that the agreement cannot transition out of terminal states. Since each transition requires the agreement to be **Active**, and each transition sets a distinct terminal state, only one terminal transition can succeed. Thus, settlement is deterministic and non duplicative.

7.8 Property 6: Encumbrance Preservation Under Direct Offers and Agreements

Claim. While an offer is open or an agreement is active, the principal and collateral committed to it remain non withdrawable and non reusable.

Proof sketch. Open offers introduce an encumbrance at posting time, either as $\text{offerEscrow}_{L,P_p}$ for lend offers or as locked_{B,P_c} for borrow offers. Acceptance converts the relevant encumbrance into active agreement encumbrances and direct principal state variables. Because withdrawals reference the encumbrance inequality in Property 2, funds committed to offers and agreements cannot be withdrawn. Because posting new offers and drawing on credit also reference available unencumbered balances, the same encumbrance variables prevent reuse. Cancellation and settlement remove encumbrances atomically in the same transaction that ends the obligation.

7.9 Property 7: Collateral Continues Earning Without Breaking Safety

Claim. A borrower can lock collateral for a direct agreement without removing it from pool accounting, enabling continued index based accrual, while still preventing withdrawal of that collateral.

Proof sketch. Locking collateral updates locked_{B,P_c} but does not reduce dep_{B,P_c}. Fee and maintenance accrual are computed from bases that are functions of dep_{B,P_c} and related variables. Therefore, the collateral can remain part of accrual computation. Simultaneously, withdrawability is reduced because locked_{B,P_c} appears in Equation 5. Thus, earning and withdrawal safety are compatible because accrual and withdrawability are decoupled by design.

7.10 Property 8: Revolving Credit Schedule Enforcement

Claim. For a revolving credit line, required periodic payments are deterministically enforceable and delinquency can be represented as explicit state that restricts risk increasing actions.

Proof sketch. Revolving credit introduces explicit schedule state variables such as $\mathrm{nextDue}_{x,P}$ and $\mathrm{arrears}_{x,P}$. At any transition that would increase debt or decrease secured buffer, the protocol checks whether the position is delinquent, for example whether $t \geq \mathrm{nextDue}_{x,P}$ and the required due is unpaid. If delinquent, the transition reverts or is restricted. Payment transitions update $\mathrm{arrears}_{x,P}$ and advance $\mathrm{nextDue}_{x,P}$ deterministically. Because these checks are local and time based, enforcement does not depend on external prices or liquidations.

7.11 Property 9: Gas Boundedness of Core Operations

Claim. Core operations are bounded in computational complexity with respect to the number of positions and agreements, and are feasible under typical block gas limits.

Proof sketch. Each core operation updates a constant number of state variables for a constant number of pools and positions, and does not iterate over global sets. Offer books are implemented as append only records with direct lookup by identifiers for acceptance and cancellation. As a result, gas costs scale with the number of storage writes in the specific operation, not with total protocol usage.

7.12 Discussion and Limits of Proof Sketches

These sketches assume correct implementation of guards, correct handling of non standard token behavior via allowlisting or adapters, and correct enforcement of non reentrancy where external calls occur. They also do not claim economic optimality. In particular, while oracle free enforcement avoids liquidation cascades, it shifts risk evaluation to counterparties selecting terms and collateral. The properties above assert correctness of state transitions and isolation of losses, not that any given term is rational.

Section 8 expands on attack surfaces that could violate assumptions.

8 Security Analysis

This section analyzes the main attack surfaces of EQUALIS and the conditions under which the properties in Section 7 could fail. The protocol is designed to remove oracle and liquidation specific failure modes, but it still inherits the usual EVM hazards and introduces new mechanism specific risks, particularly around encumbrance accounting, offer book griefing, and timing boundary games.

8.1 Threat Surface Summary

We group threats into six buckets:

- token behavior hazards and accounting drift,
- reentrancy and external call control flow,
- MEV and transaction ordering,
- encumbrance accounting bugs and invariant violations,
- offer book griefing and storage growth,
- parameterization and governance risk.

8.2 Token Behavior Hazards

The protocol's strongest accounting claims assume supported tokens behave like conventional ERC20. In practice, many do not.

Fee on transfer and deflationary tokens. If a token takes a fee on transfer, then the amount credited by the protocol and the amount received by the pool diverge. This breaks pool reconciliation, corrupts indices, and may create phantom credit.

Rebasing tokens. Rebasing changes balances without explicit transfers. Any invariant expressed as equality between trackedBalance_P and the token balance can fail. In addition, fee and maintenance bases derived from principal accounting no longer map cleanly to real balances.

Non standard return behavior. Some tokens do not revert on failure, or return false. The protocol must use safe transfer wrappers that treat any failure as a revert, and must not assume success based on return value presence alone.

Callback hooks and ERC777 like behavior. Tokens with callbacks can reenter the protocol during transfers. Even standard ERC20 can be wrapped by a contract that introduces unexpected call patterns. Therefore, any external transfer must be considered a reentrancy boundary.

Mitigation. The protocol should default to allowlisting supported tokens, or implement adapters for problematic tokens. For unsupported tokens, claims in Sections 4 and 7 do not hold without additional assumptions.

8.3 Reentrancy and External Calls

Reentrancy targets. The most damaging reentrancy targets are functions that:

- update encumbrance variables,
- transfer tokens,

- update index snapshots or bases,
- create or settle agreements.

A reentrant call that executes before state is fully updated could allow double withdrawal, double settlement, or encumbrance bypass.

Control flow discipline. The protocol should enforce:

- checks effects interactions, with state updates completed before any external transfer,
- explicit non reentrancy guards on functions that cross trust boundaries,
- avoidance of external calls in the middle of multi pool updates where partial state could be exploited.

Facet and modular reentrancy. In a modular architecture, cross facet calls are internal, but external calls can occur inside one facet and reenter through another. Guards must be global at the diamond level, not local per facet, or the protocol must ensure no external call can occur before all invariants are restored.

8.4 Flash Loan Attack Surface

Flash loans introduce an explicit external callback boundary where arbitrary receiver code executes between loan disbursement and repayment verification. This creates a distinct attack surface that warrants dedicated analysis.

Reentrancy boundary. The flash loan callback receiver can attempt to call back into any protocol function, including deposit, withdraw, borrow, and offer acceptance. The protocol enforces a diamond level `nonReentrant` modifier on the flash loan function, which prevents reentry into any other guarded function during callback execution. This is critical because partial state (tokens transferred out, balance not yet verified) exists during the callback window.

Same transaction state manipulation. An adversary with flash loaned funds can attempt to manipulate protocol state within the same transaction before repayment. Potential vectors include:

- depositing flash loaned funds to inflate fee bases or index snapshots,
- accepting offers using flash loaned liquidity,
- triggering index updates or maintenance settlements.

The reentrancy guard prevents direct manipulation during the flash loan callback. However, if the receiver contract calls external protocols that then call back into EQUALIS, the guard must cover all entry points.

Invariant restoration. The flash loan function follows checks effects interactions by verifying repayment via explicit balance check after the callback completes. The function pulls repayment explicitly from the receiver rather than relying on the receiver to push tokens, preventing cross pool balance spoofing where an attacker might transfer tokens to a different pool's address.

Fee on transfer and rebasing token incompatibility. Flash loans inherit the same token behavior risks as other pool operations. If a pool's underlying token takes fees on transfer, the amount received by the receiver and the amount pulled back will differ from expected values, potentially causing the repayment check to fail or to accept insufficient repayment. Rebasing tokens can change balances between the pre callback snapshot and post callback verification. Flash loans should only be enabled for pools with allowlisted, well behaved tokens.

Anti split protection. The optional $\text{flashLoanAntiSplit}_P$ flag prevents a receiver from executing multiple flash loans in the same block for the same pool. This mitigates attacks that split a large loan into smaller loans to reduce cumulative fees or to interleave state changes between loans.

Griefing via callback reversion. A receiver can intentionally revert during the callback to waste the caller's gas. This is expected behavior and does not affect protocol correctness. The protocol does not attempt to prevent callback reversion griefing, as doing so would require removing the atomicity guarantee that makes flash loans useful.

8.5 MEV and Ordering Games

Offer acceptance races. Because offers are executable by anyone, multiple parties may attempt to accept the same offer in the same block. Correctness requires that only the first acceptance succeeds and that later attempts revert cleanly without partial side effects.

Boundary time games. Direct agreements are time gated. MEV searchers may attempt to reorder repayment and recovery transactions near t_r. The protocol should define strict inequality conditions such that the set of valid callers is disjoint before and after t_r. If both repayment and recovery could succeed in the same timestamp regime, ordering could determine who receives collateral.

Index update front running. Index based yield accrual implies that users might attempt to time deposits or withdrawals around fee distribution updates. This is not a correctness failure, but it is an economic surface. If index updates occur only on interactions, a user could attempt to force settlement ordering to capture more yield. The protocol should treat this as expected behavior under an index model and ensure that fee bases and snapshots update deterministically per user interaction.

8.6 Encumbrance Accounting Bugs

Encumbrance accounting is the core safety mechanism. The most catastrophic class of bugs is one that allows a user to withdraw or reuse encumbered funds.

Missing encumbrance terms. If withdrawability checks omit a term, for example failing to include $\text{offerEscrow}_{x,P}$ for lender posted offers or failing to include collateral locks for borrower posted offers, users could withdraw funds that are still required for agreement settlement.

Inconsistent conversion at acceptance. Offer acceptance converts open offer encumbrances into active agreement encumbrances. If this conversion is not atomic and correct, the protocol could double count or fail to count encumbrances, allowing either withdrawal of committed funds or perpetual lock of funds after agreement settlement.

Cross pool mismatch. Direct agreements span two pools. Any mismatch of pool identifiers in storage updates could lock the wrong asset or release the wrong encumbrance, violating pool isolation and potentially leading to loss.

Mitigation. The protocol should treat encumbrance invariants as testable properties:

- for every transition, assert that withdrawability remains non negative,
- for every direct agreement transition, assert encumbrance deltas sum to zero in the relevant pools,
- for every settlement path, assert terminal states are single use and that collateral is transferred exactly once.

Property based testing is particularly appropriate because random sequences of offers, accepts, cancels, repayments, exercises, and recoveries can reveal missing guards.

8.7 Offer Book Griefing and Storage Growth

Spam offers. Because offers can be posted permissionlessly, an adversary can create many offers to bloat storage and indexing load. While this does not break correctness, it can degrade UX and increase indexer costs.

Mitigation options. Possible mitigations include:

- configurable per position offer caps,
- posting fees that are routed into pool fee distribution,
- minimum offer sizes per pool,
- expiration times that allow pruning or lazy invalidation.

These mitigations are policy choices. The core correctness properties do not require them, but practical deployment may.

8.8 Timing and Liveness Risks for Revolving Credit

Revolving credit lines introduce periodic payment state. A common failure mode is ambiguity around what happens when a payment is overdue.

Delinquency representation. Delinquency must be represented explicitly, for example via $\text{arrears}_{x,P}$ and an overdue $\text{nextDue}_{x,P}$, and must restrict risk increasing actions deterministically. If delinquency is only inferred, edge cases can appear where draws or withdrawals slip through.

Grace and tolerance parameters. If tolerance windows are too short, liveness suffers because minor timestamp variance can cause unexpected delinquency. If tolerance windows are too long, lenders face delayed enforcement. These are parameterization tradeoffs rather than correctness issues, but they affect protocol safety perception.

8.9 Parameterization and Governance Risk

Even in an oracle free design, configuration can introduce risk.

Penalty rates and settlement rules. Bounded default penalties are a design choice. If penalties are too low, the system may incentivize strategic default. If too high, the system becomes punitive and loses the advantage over liquidation based designs. The protocol should treat penalty parameters as explicit, reviewable values with documented rationale.

Fee and maintenance rates. Fee and maintenance parameters affect both yield and the cost of holding encumbered collateral. Because locked collateral can continue earning, excessive maintenance rates could negate the borrower advantage and change economic incentives.

Admin key risk. Any privileged ability to modify parameters or pause functionality is a centralized trust surface. If present, it should be minimized, time delayed, and transparently auditable. This paper assumes normal operation does not require privileged intervention.

8.10 Economic Risks Not Covered by Correctness Proofs

Counterparty selection. Direct agreements shift the burden of risk evaluation to counterparties. The protocol ensures deterministic settlement, but it does not guarantee that a given collateral and term pair is rational.

Strategic behavior. Because borrowers can choose between repayment and collateral forfeiture, the agreement resembles an embedded option. This is intended, but it means lenders must price the premium accordingly. Incorrect pricing is an economic failure, not a protocol correctness failure.

Collateral yield exploitation. Collateral that continues earning is advantageous to borrowers. However, it can also create complex incentives if fee base normalization is weak or if locked collateral earns in a way that can be recursively amplified. This motivates the normalized fee base and explicit base definitions. The protocol should test for looping strategies that attempt to increase fee base without net contribution.

8.11 Summary

Equalis removes a class of failures associated with liquidation cascades and oracle dependence, but it replaces them with a smaller set of sharp requirements: strict encumbrance accounting, deterministic time gated state machines, and disciplined handling of external token transfers. These requirements are amenable to property based testing and invariant driven audits. The next section reports gas usage measurements to ground practical evaluation of the protocol as deployed code.

9 Related Work

This section situates Equalis relative to common onchain lending patterns and adjacent designs. The goal is not an exhaustive survey, but to clarify which assumptions are shared, which are rejected, and which tradeoffs are intentionally chosen.

9.1 Liquidation Based Over Collateralized Lending

Most deployed DeFi money markets use over collateralization with liquidation based enforcement. Borrowers post collateral and may borrow up to an LTV threshold. If collateral value falls relative to debt, liquidators can repay debt and seize collateral at a discount. This pattern couples solvency to continuous market repricing and to oracle availability.

The Equalis pooled credit primitive differs in two ways. First, pooled enforcement does not depend on price thresholds or collateral auctions. Second, cross asset credit is not expressed as a pool wide exposure; it is expressed only through bilateral direct agreements. The intent is to remove liquidation cascades and to avoid protocol correctness depending on market depth at the moment enforcement is triggered.

9.2 Oracle Reliant Cross Asset Lending

Cross asset borrowing in DeFi typically relies on price oracles to determine whether a position is solvent and to set liquidation thresholds. This introduces oracle risk, including incorrect pricing, liveness failures, and manipulation.

Direct eliminates oracle dependence by replacing continuous solvency checks with deterministic settlement. Parties negotiate terms and collateral ex ante, and the protocol enforces settlement via time gated transitions. This does not remove economic risk. It relocates it to counterparties, who must price the embedded option like structure of repayment versus forfeiture.

9.3 Peer to Peer Lending and Order Book Models

Peer to peer lending onchain appears in two broad forms.

P2P matching over pooled liquidity. Some systems match lenders and borrowers but still rely on pooled liquidity as a source of truth for rate setting or enforcement. Others use pooled liquidity for settlement while exposing users to pool wide risks.

Order book style term discovery. A smaller class of designs treat loan terms as offers that can be posted and taken, enabling explicit rate discovery. Equalis extends this by allowing both sides to post executable offers. Lender posted offers escrow principal. Borrower posted offers lock collateral. This symmetry creates a loan offer book that resembles a CLOB for loan terms while preserving deterministic settlement.

9.4 Fixed Term and Revolving Credit

Most DeFi lending is effectively open ended debt whose enforcement is liquidation based rather than schedule based. Fixed term loans exist in some protocols, but they commonly remain tied to oracle based solvency and liquidation semantics.

Traditional finance distinguishes between term loans and revolving credit lines with periodic payments. Equalis adopts this product distinction onchain. Term borrows have a maturity based settlement rule. Revolving credit introduces explicit schedule state and delinquency representation. Enforcement is time based rather than price reactive.

9.5 Options Like Credit Agreements

A secured loan with the borrower having the choice to repay principal or forfeit collateral is economically similar to an embedded option. In liquidation based systems, this optionality is mediated by market participants through liquidation auctions and discount mechanisms. In Equalis Direct, the optionality is explicit and is priced via the premium π paid at acceptance.

This explicit optionality clarifies risk allocation. Lenders must price the premium given the collateral and the borrower's outside opportunities. Borrowers can choose early exercise by forfeiting collateral. The protocol's role is not to decide the price, but to enforce the resulting contract deterministically.

9.6 Index Products and Fee Distribution

Index token systems onchain typically implement basket custody, mint and burn, and fees. A common implementation detail is whether fees are extracted by directly reducing vault balances or by maintaining explicit fee pots.

EqualIndex adopts an explicit fee pot model that separates vault custody balances from distributable fee balances. This is conceptually consistent with Equalis's preference for explicit encumbrance and explicit accounting rather than implicit balance deltas. The two systems can compose at the asset layer, subject to token behavior constraints and allowlisting.

9.7 Summary of Contrasts

Relative to the dominant liquidation based DeFi lending pattern, Equalis makes the following deliberate departures:

- **No reactive liquidations.** Enforcement is time and rule based, not price threshold based.
- **No oracle dependence for correctness.** Cross asset credit is expressed via bilateral contracts rather than pool wide health factors.
- **Explicit offer book for terms.** Either side can post executable offers, enabling direct rate discovery.
- **Collateral remains economically active.** Borrower locked collateral can continue to accrue pool yield while remaining non withdrawable.
- **Product distinction.** Term loans and revolving credit lines are separate primitives with distinct enforcement rules.

These contrasts define the main tradeoff surface: Equalis reduces liquidation and oracle failure modes, but requires counterparties to price bilateral risk and requires strict encumbrance accounting to maintain deterministic safety properties.

10 Limitations and Future Work

This paper describes an oracle free, non reactive lending design. The design removes a class of liquidation and oracle failure modes, but it also introduces limitations and open questions. This section states them directly and outlines future work that would make the system more complete and more robust.

10.1 Limitations

10.1.1 Economic Risk Is Shifted, Not Eliminated

Direct removes oracle dependence by enforcing settlement via time and explicit rules. This shifts risk evaluation to counterparties. Lenders must price the premium π given collateral, maturity, and the borrower's optionality to repay or forfeit. Incorrect pricing can lead to economic losses even if protocol correctness holds.

Similarly, pooled credit that is self secured avoids liquidation cascades, but it does not guarantee that any given credit term is economically rational for a user. The protocol enforces rules, not outcomes.

10.1.2 Bounded Penalty Design Is Coarse

Bounded, rules based default settlement is intentionally simpler than liquidation auctions, but it is also less expressive. A fixed penalty rate is a blunt instrument. If too low, strategic default becomes attractive. If too high, the protocol becomes punitive and loses its primary advantage over liquidation based designs.

A further limitation is that penalty rules are a policy decision that may need iterative tuning. Without careful parameter governance, bounded settlement can degrade either to trivial enforcement or to de facto liquidation severity via penalty magnitude.

10.1.3 Revolving Credit Requires Careful Schedule Semantics

Revolving credit lines introduce periodic payments and delinquency state. Small ambiguities around when a payment becomes due, how arrears are computed, and what actions are restricted can create edge cases that are exploitable or confusing.

The protocol can make schedule rules deterministic, but it cannot prevent user confusion if the rules are not legible. Clear UI and clear event emissions are required for users to understand their next due, arrears, and restricted actions.

10.1.4 Token Compatibility and Allowlisting Remain Necessary

Even though the protocol is oracle free, it is not token agnostic in practice. Fee on transfer tokens, rebasing tokens, and callback heavy tokens can violate reconciliation assumptions and create reentrancy risk. Allowlisting and or adapters are still required for robust deployment.

This limitation is structural to onchain systems that rely on balance accounting. It is not unique to Equalis, but it must be stated because it affects composition.

10.1.5 Offer Book Griefing and Indexing Load

A permissionless offer book can be spammed. The protocol can remain correct under spam, but indexers and UIs may degrade. Storage growth is also permanent on most L1 settings, so spam can impose long run costs.

Mitigations such as posting fees, minimum sizes, expirations, and per position caps are policy choices that trade openness for practical operability.

10.1.6 Gas and UX Constraints

Measured gas usage is bounded, but some operations are storage heavy, particularly those that create or activate multi party objects. This affects UX on L1 during high fee regimes. The paper reports gas units, but cost sensitivity remains a practical limitation for adoption.

10.2 Future Work

10.2.1 Formalization and Mechanized Verification

Section 7 provides informal proof sketches. Future work includes:

- a full transition system specification with explicit preconditions and postconditions,
- mechanized verification of encumbrance invariants, terminal state uniqueness, and pool isolation,
- systematic exploration of rounding behavior and integer overflow protections.

10.2.2 Fee Base and Maintenance Base Policy Exploration

Collateral continuing to earn is a core feature, but its net effect depends on fee base and maintenance base definitions. Future work should evaluate:

- whether locked collateral contributes fully, partially, or not at all to the fee base,
- whether maintenance charges should be higher for encumbered balances to account for liquidity constraints,
- whether base definitions can be tuned to reduce looping incentives without reducing legitimate borrower benefit.

This is a mechanism design problem. The protocol can support multiple policies, but the most robust policy should be informed by adversarial simulation and empirical usage.

10.2.3 More Expressive Credit Products

The system currently supports pooled term loans, pooled revolving credit lines, and bilateral direct agreements. Future work could include:

- amortizing term structures with periodic principal and interest schedules,
- partial repayment and partial collateral release rules for direct agreements,
- multi collateral direct agreements,
- portfolio level constraints for positions spanning multiple pools.

The challenge is to extend expressiveness without reintroducing reactive liquidation semantics.

10.2.4 Secondary Markets for Positions and Agreements

Position NFTs enable transfer of account state. Future work includes:

- standardized markets for position transfers with clear pricing of embedded obligations,
- risk disclosure primitives for positions with active direct agreements or revolving debt,
- permissionless settlement tooling for buyers of positions to manage obligations post transfer.

This is likely essential for liquidity and for enabling specialized underwriters and traders to take over positions from retail borrowers.

10.2.5 Anti Griefing and Market Quality Controls

Practical deployments may require market quality features:

- expirations and pruning rules for offers,
- maker and taker fees that fund indexers and spam resistance,
- reputation or stake weighted offer visibility as a UI layer, not a protocol requirement.

The protocol should remain neutral, but the ecosystem can evolve defenses at the application layer.

10.2.6 Deployment Architecture and Scaling

Because the design is deterministic and does not rely on oracle updates, it may be well suited to L2 deployment environments where fees are lower and latency is lower. Future work includes:

- evaluating L2 deployment tradeoffs, including bridge risk and sequencing risk,
- optimizing storage layouts and event schemas for indexers,
- exploring batch operations for common workflows, such as accept plus optional yield settlement.

10.3 Summary

Equalis trades reactive liquidations for deterministic settlement, and oracle dependence for bilateral pricing. The primary limitations arise from economic pricing, parameter tuning, token behavior constraints, and offer book operability. Future work should focus on mechanized verification, robust base policy definitions, richer credit products, and ecosystem tooling to support markets for positions and agreements.

11 Conclusion

This paper presented Equalis, a lending protocol designed around deterministic settlement and non reactive risk semantics. The system intentionally avoids liquidation based enforcement and oracle dependence for correctness. Instead, it constrains outcomes through explicit encumbrance accounting, time gated state machines, and localized pool scoped invariants.

Equalis combines two complementary primitives. The pooled credit layer provides single asset term loans and self secured revolving credit lines with explicit periodic payment state and deterministic delinquency enforcement. The Direct layer provides oracle free cross asset bilateral credit agreements implemented as an executable loan offer book where either side can post offers. Direct agreements settle through repayment, borrower exercise, or lender recovery after a grace period. A distinguishing feature is that borrower collateral remains within pooled accounting and can continue to accrue pool yield while remaining non withdrawable, allowing borrowers to keep capital productive while deploying borrowed funds externally.

We stated core correctness properties and provided proof sketches for reconciliation, encumbrance safety, pool isolation, non socialized loss, deterministic settlement, and schedule enforcement. We also reported measured gas usage for core operations to support reproducible evaluation.

The central tradeoff of this design is that it relocates risk. By removing reactive liquidations and oracle dependence, Equalis reduces a class of failure modes associated with cascades and pricing feeds. In exchange, it requires strict accounting correctness and it places greater responsibility on counterparties to price bilateral terms, including the embedded optionality in direct agreements. This is an explicit design choice, consistent with the goal of making credit outcomes legible and bounded at origination rather than emergent from market microstructure at liquidation time.

Future work includes mechanized verification of the stated invariants, empirical evaluation of fee and maintenance base policies, richer credit products such as amortization schedules, and ecosystem tooling for secondary markets in Position NFTs and agreement states. Together, these directions can strengthen the thesis that deterministic, non reactive credit primitives can be a viable alternative foundation for onchain lending.

A Glossary

Pool A single asset accounting domain parameterized by an ERC20 like token $\tau(P)$.

Position NFT A transferable account container that owns pool scoped balances, encumbrances, and obligations.

Encumbrance Any balance component that is not withdrawable, represented explicitly (for example locked collateral or offer escrow).

Term Loan A pooled borrow with an explicit maturity rule and bounded default settlement.

Revolving Credit Line A pooled borrow facility with redraw capability and deterministic periodic payment requirements.

Flash Loan An atomic borrow and repay operation that completes within a single transaction, requiring repayment plus fee before transaction end; does not create position level debt.

Delinquency A state in which required periodic payments are overdue, represented explicitly (for example via arrears).

Direct Agreement A bilateral credit agreement between two positions, possibly cross asset, enforced without price oracles.

Lend Offer A direct offer posted by a lender that escrows principal and can be accepted by a borrower.

Borrow Offer A direct offer posted by a borrower that locks collateral and can be accepted by a lender.

Premium Interest paid at acceptance for a direct agreement, realized upfront and not refunded on early repayment.

Fee Index Pool level index that distributes system generated fees to positions according to a normalized fee base.

Maintenance Index Pool level index that applies proportional maintenance charges over time according to a maintenance base.

Fee Pot A balance reserved for fee distribution that is tracked explicitly, for example in EqualIndex.

B Notation and State Variables

B.1 Sets and Identifiers

- $\mathcal{T}$: set of supported tokens.
- P: a pool, with underlying token $\tau(P) \in \mathcal{T}$.
- x: a position identifier (Position NFT).
- L: lender position identifier in a direct agreement.
- B: borrower position identifier in a direct agreement.
- A: a direct agreement identifier.
- O: a direct offer identifier.

B.2 Per Pool Variables

- trackedBalance_P: internal accounting of pool token balance.
- I_P^{fee}: fee distribution index.
- I_P^{mtn}: maintenance charging index.
- flashLoanFeeBps_P: flash loan fee rate in basis points.
- $\text{flashLoanAntiSplit}_P$: boolean flag enabling per block anti split protection.

B.3 Per Position, Per Pool Variables

- $\text{dep}_{x,P}$: deposited principal credited to position x in pool P.
- $\text{debt}_{x,P}$: pooled debt owed by x to pool P.
- $\text{locked}_{x,P}$: locked principal in pool P (collateral locks and similar).
- $\text{offerEscrow}_{x,P}$: principal escrowed to fund open direct offers.
- $\text{directLent}_{x,P}$: principal lent via active direct agreements.
- $\text{directBorrowed}_{x,P}$: principal borrowed via active direct agreements.
- $B_{x,P}^{\text{fee}}$: normalized fee base for fee index accrual.
- $B_{x,P}^{\text{mtn}}$: maintenance base for maintenance index charging.
- $I_{x,P,\text{snap}}^{\text{fee}}$: last settled fee index snapshot for x in P.
- $I_{x,P,\text{snap}}^{\text{mtn}}$: last settled maintenance index snapshot for x in P.

B.4 Revolving Credit Line Variables

- $\text{nextDue}_{x,P}$: timestamp for the next required periodic payment.
- $\text{arrears}_{x,P}$: accumulated unpaid due amounts.
- $\text{apr}_{x,P}$: interest rate parameter for schedule computation, if applicable.

B.5 Direct Agreement Parameters

- p: principal amount (asset τ_p in principal pool P_p).
- c: collateral amount (asset τ_c in collateral pool P_c).
- π: premium paid at acceptance (typically in principal asset).
- t_m: maturity timestamp.
- Δ_g: grace period duration.
- $t_r = t_m + \Delta_g$: recovery time.

C Key Constraints and Identities

C.1 Withdrawability Constraint

For any position x and pool P, define:

$$\text{withdrawable}_{x,P} = \text{dep}_{x,P} - \Big(\text{locked}_{x,P} + \text{offerEscrow}_{x,P} + \text{directLent}_{x,P}\Big). \tag{51}$$

A withdrawal of amount a is valid only if:

$$a \le \text{withdrawable}_{x,P}. \tag{52}$$

C.2 Recovery Time

For a direct agreement A with maturity t_m and grace Δ_g:

$$t_r = t_m + \Delta_g. \tag{53}$$

C.3 Fee Accrual Identity

For pool P and position x:

$$\text{feeAccrued}_{x,P} = B^{\text{fee}}_{x,P} \cdot \left(I^{\text{fee}}_P - I^{\text{fee}}_{x,P,\text{snap}}\right). \tag{54}$$

C.4 Maintenance Owed Identity

For pool P and position x:

$$\text{mtnOwed}_{x,P} = B^{\text{mtn}}_{x,P} \cdot \left(I^{\text{mtn}}_P - I^{\text{mtn}}_{x,P,\text{snap}}\right). \tag{55}$$

D Direct Offer and Agreement State Machines

D.1 Offer Lifecycle

Table 1: Direct offer lifecycle.

State	**Transition**	**Preconditions**
Open	Cancelled	Caller is poster, offer not filled
Open	Filled	Counterparty acceptance succeeds
Cancelled	(terminal)	None
Filled	(terminal)	Agreement created

D.2 Agreement Lifecycle

Table 2: Direct agreement terminal transitions.

Transition	**Caller**	**Time Condition**
Active $\to$ Repaid	Borrower	$t < t_r$
Active $\to$ Exercised	Borrower	$t < t_r$
Active $\to$ Recovered	Lender	$t \ge t_r$

D.3 Encumbrance Deltas Summary

Table 3: Encumbrance deltas for direct offer and agreement actions.

Action	**Encumbrance delta**
Post lend offer	$\text{offerEscrow}_{L,P_p} \mathrel{+}= p$
Post borrow offer	$\text{locked}_{B,P_c} \mathrel{+}= c$
Cancel lend offer	$\text{offerEscrow}_{L,P_p} \mathrel{-}= p$
Cancel borrow offer	$\text{locked}_{B,P_c} \mathrel{-}= c$
Accept lend offer	$\text{offerEscrow}_{L,P_p} \mathrel{-}= p$, $\text{directLent}_{L,P_p} \mathrel{+}= p$, $\text{directBorrowed}_{B,P_p} \mathrel{+}= p$, $\text{locked}_{B,P_c} \mathrel{+}= c$
Accept borrow offer	$\text{directLent}_{L,P_p} \mathrel{+}= p$, $\text{directBorrowed}_{B,P_p} \mathrel{+}= p$
Repay	$\text{directBorrowed}_{B,P_p} \mathrel{-}= p$, $\text{directLent}_{L,P_p} \mathrel{-}= p$, $\text{locked}_{B,P_c} \mathrel{-}= c$
Exercise	$\text{directBorrowed}_{B,P_p} \mathrel{-}= p$, $\text{directLent}_{L,P_p} \mathrel{-}= p$, $\text{locked}_{B,P_c} \mathrel{-}= c$
Recover	$\text{directBorrowed}_{B,P_p} \mathrel{-}= p$, $\text{directLent}_{L,P_p} \mathrel{-}= p$, $\text{locked}_{B,P_c} \mathrel{-}= c$

E Revolving Credit Schedule Semantics

E.1 Deterministic Delinquency Flag

A position x in pool P is delinquent if:

$$t \geq \text{nextDue}_{x,P} \quad \wedge \quad \text{arrears}_{x,P} > 0. \tag{56}$$

While delinquent, actions that increase risk should be restricted, such as new draws and withdrawals that reduce secured buffer. Payment transitions should update arrears and advance the schedule deterministically:

$$\text{arrears}_{x,P} \leftarrow \max\{0, \text{arrears}_{x,P} - \text{paid}\}, \qquad \text{nextDue}_{x,P} \leftarrow \text{nextDue}_{x,P} + \Delta_{\text{period}}. \tag{57}$$

E.2 Periodic Due Computation Placeholder

The protocol can define the periodic due as a function of outstanding debt and time in the period. This paper treats the due function as a configurable product policy. Implementations should specify:

- the base used for the due (for example average debt, end of period debt, or maximum debt),
- rounding behavior and minimum payment rules,
- whether due includes principal amortization, interest only, or both.

F Property Checklist for Audits and Tests

This checklist is intended to map the claims in Section 7 into testable assertions.

1. Pool reconciliation holds for supported tokens after every deposit, withdraw, borrow, repay, flash loan, and direct settlement transition.
2. Flash loan atomicity: if repayment is insufficient, the transaction reverts with no state changes; if successful, trackedBalance_P increases by exactly the fee amount.

3. Flash loans do not modify any position state variables (no debt created, no encumbrances changed).
4. Flash loan reentrancy guard prevents callback from reentering any guarded protocol function.
5. Anti split protection (when enabled) prevents multiple flash loans from the same receiver in the same block.
6. Withdrawability never becomes negative for any position and pool.
7. Offer posting increases encumbrances and cancellation decreases encumbrances atomically.
8. Offer acceptance converts open encumbrances to active encumbrances without leakage or double counting.
9. Direct agreements reach exactly one terminal state and cannot be settled twice.
10. Before t_r, lender recovery is impossible. After t_r, borrower repayment and exercise are impossible.
11. Pool isolation holds: no operation modifies position state in a pool not referenced by the call parameters.
12. Locked collateral continues to accrue according to index base policy while remaining non withdrawable.
13. Revolving delinquency state restricts new draws and restricts withdrawals that reduce secured buffer.
14. No function performs unbounded iteration over global sets.