CapSign Protocol

A Hierarchical Ownership Graph for Programmable Capital Markets

Version 1.0

CapSign, Inc.

February 2026

Abstract

CapSign Protocol introduces a novel architecture for representing ownership, capital, and financial obligations as a hierarchical directed acyclic graph (DAG) on Ethereum-compatible blockchains. Unlike traditional token standards that model assets as isolated contracts, CapSign constructs a unified *ownership graph* where entities, wallets, tokens, and lots form an interconnected structure that mirrors real-world corporate hierarchies and investment relationships. This architecture enables automatic balance sheet generation, real-time settlement with instant finality, regulatory-compliant token transfers, and sophisticated capital operations including capital calls, distributions, and secondary transfers with right of first refusal. We present the formal model, smart contract architecture, and demonstrate how this approach reduces operational overhead by orders of magnitude while increasing transparency and reducing settlement risk to zero.

Contents

1 Introduction

The infrastructure underpinning private capital markets remains fragmented, opaque, and operationally burdensome. Fund administrators, transfer agents, and legal counsel spend countless hours reconciling cap tables, tracking beneficial ownership, processing capital calls, and generating investor statements. Meanwhile, investors often wait weeks for settlements, lack real-time visibility into their holdings, and face significant friction when seeking liquidity through secondary sales.

CapSign Protocol addresses these challenges by reconceptualizing ownership not as a set of isolated token balances, but as a *hierarchical ownership graph*—a structured representation of who owns what, through which entities, subject to what restrictions, and with what associated rights and obligations.

1.1 Design Principles

The protocol is built on four foundational principles:

1. **Hierarchical Composability**: Ownership naturally forms hierarchies. An individual owns shares in an LLC, which owns units in a fund, which owns equity in portfolio companies. The protocol must represent these nested structures natively.
2. **Full Balance Sheet**: Assets and liabilities are two sides of the same coin. A token representing a receivable is simultaneously an asset to the holder and a liability to the issuer. The protocol maintains this duality.
3. **Automatic Ledgering**: Every on-chain action generates an auditable journal entry. The ledger is not a separate system to be reconciled—it *is* the chain state.
4. **Real-Time Settlement**: Transactions settle atomically at execution time. There is no T+1, T+2, or T+N. Finality is immediate.

2 The Hierarchical Ownership Graph

2.1 Formal Model

Let $G = (V, E)$ be a directed acyclic graph where:

- $V = \mathcal{E} \cup \mathcal{W} \cup \mathcal{T} \cup \mathcal{L}$ is the vertex set, partitioned into:
  - $\mathcal{E}$: Entities (individuals, corporations, trusts, funds)
  - $\mathcal{W}$: Wallets (smart contract accounts)
  - $\mathcal{T}$: Tokens (fungible and non-fungible instruments)
  - $\mathcal{L}$: Lots (specific tranches of token holdings)
- $E \subseteq V \times V$ is the edge set representing ownership, control, and delegation relationships

For any vertex $v \in V$, we define:

- parent(v): The immediate owner or controller of v
- children(v): The set of vertices owned or controlled by v
- ancestors(v): The transitive closure of parent
- descendants(v): The transitive closure of children

Type	Description	Examples
INDIVIDUAL	Natural person	Investor, founder
CORPORATION	C-Corp or S-Corp	Delaware C-Corp
LLC	Limited liability company	Fund vehicle
TRUST	Legal trust structure	Family trust
PARTNERSHIP	LP or GP	Fund GP
FUND	Investment vehicle	VC fund, PE fund
DAO	Decentralized organization	Wyoming DAO LLC

Table 1: Owner types in the CapSign Protocol

2.2 Owner Hierarchy

Owners form the root of ownership chains. The ownership graph is agnostic to owner type—individuals, legal entities, trusts, and any other organizational form can participate. Table 1 shows common owner types, but the protocol imposes no constraints on what can be represented.

For legal entity classification, the protocol uses ISO 20275 Entity Legal Form (ELF) codes, maintained by GLEIF. This international standard provides identifiers for legal entity types across jurisdictions. For emerging structures not yet assigned ELF codes—such as Wyoming DAO LLCs or other novel formations—we define provisional codes and aim to contribute them to the standard as these structures gain regulatory recognition. Individuals are classified separately as natural persons. This approach ensures interoperability with global financial infrastructure while accommodating both traditional and novel ownership structures.

Each entity is associated with exactly one *primary wallet*—a smart contract account that holds the entity's on-chain assets and executes transactions on its behalf.

2.3 Wallet Architecture

Wallets in CapSign are not simple externally-owned accounts (EOAs). They are *diamond proxies* implementing EIP-2535, enabling modular functionality through composable facets:

```
WalletCoreFacet      // Identity, metadata, ownership
WalletSignatureFacet // EIP-1271 signature validation
AccessManagedFacet   // Role-based permissions
TokenHolderFacet     // Receive and manage tokens
CapitalCallFacet     // Fund capital operations
InvestmentFacet      // Portfolio tracking
DocumentsFacet       // On-chain document registry
```

Listing 1: Core wallet facets

This architecture enables wallets to evolve over time. A startup's wallet may initially include basic token issuance capabilities, then add cap table management, then compliance modules, then liquidity features—all without deploying a new contract or migrating assets.

2.3.1 Wallet Ownership

Each smart wallet has one or more *owners* who can authorize transactions. Owners can be:

- **Passkeys**: WebAuthn credentials backed by device biometrics (Face ID, Touch ID, Windows Hello) and cloud-synced across devices. This enables a fully self-custodial experience without seed phrases.
- **EOAs**: Traditional Ethereum accounts controlled by private keys. These may be held directly by the user or managed by a qualified custodian.

- **Custodial Keys**: For institutional investors, private keys can be held within a custodian's secure infrastructure. The custodian enforces its own signing policies—approval workflows, spending limits, whitelists—within its firewall before broadcasting transactions to the network.

This flexibility allows retail investors to use familiar biometric authentication while institutions maintain their existing custody arrangements and compliance controls.

2.4 Token Model

Tokens represent ownership interests, debt obligations, or other financial instruments. Each token is itself a diamond proxy with configurable facets:

- **TokenERC20Facet**: Standard ERC-20 interface
- **TokenERC4626Facet**: Vault standard for yield-bearing tokens
- **TokenComplianceFacet**: Transfer restrictions and compliance
- **TokenLotFacet**: Tax lot tracking for cost basis
- **TokenDistributionFacet**: Dividend and distribution logic

2.5 Lot-Level Accounting

Unlike standard ERC-20 implementations that track only aggregate balances, CapSign maintains *lots*—individual tranches of token holdings with distinct metadata:

$$\text{Lot} = \langle \text{holder}, \text{token}, \text{quantity}, \text{costBasis}, \text{acquisitionDate}, \text{transferType} \rangle$$

Transfer conditions such as vesting schedules, lockups, and holding periods are attached to lots via modular compliance infrastructure rather than embedded in the lot tuple. This separation of concerns allows:

- Different compliance rules applied to different lots of the same token
- Dynamic modification of transfer conditions without reissuing lots
- Composable compliance modules that can be mixed and matched

Lot-level tracking enables:

- Specific identification for tax optimization (FIFO, LIFO, HIFO, specific lot)
- Per-lot vesting schedules
- Wash sale detection
- Vintage tracking for fund investments

3 Balance Sheet Representation

3.1 Double-Entry On-Chain

Every token transfer in CapSign implicitly represents a double-entry accounting transaction. When Entity A transfers 100 shares of Token X to Entity B:
The protocol indexes these events, enabling real-time balance sheet generation for any entity at any point in time.

Entity	Account	Debit	Credit
A	Equity Investments		100
B	Equity Investments	100	

Table 2: Automatic journal entry from token transfer

3.2 Asset Classification

Assets held by a wallet are automatically classified based on token metadata:

- **Cash & Equivalents**: Stablecoins (USDC, USDT), money market tokens
- **Investments**: Equity tokens, LP units, convertible notes
- **Receivables**: Tokenized invoices, notes receivable
- **Fixed Assets**: Tokenized real estate, equipment

3.3 Liability Tracking

The same token may represent an asset to one party and a liability to another. When a company issues debt tokens:

$$\text{For Issuer}: \text{Liabilities} \leftarrow \text{Liabilities} + \text{Principal}$$

$$\text{For Holder}: \text{Assets} \leftarrow \text{Assets} + \text{Principal}$$

The protocol maintains this duality through the `admin` field on each token, linking back to the issuing entity.

3.4 Net Asset Value Calculation

For investment vehicles, the protocol computes real-time NAV:

$$\text{NAV} = \sum_{i \in \text{Holdings}} \text{Value}(i) - \sum_{j \in \text{Liabilities}} \text{Value}(j)$$

Where Value(i) is determined by:

- Market price (for liquid tokens)
- Last round valuation (for portfolio companies)
- On-chain oracle (for yield-bearing vaults)
- Manual mark-to-market (for illiquid positions)

4 Automatic Ledgering

4.1 Event-Driven Accounting

All protocol actions emit structured events that are indexed by a subgraph:

```
event Transfer(address indexed from, address indexed to, uint256 amount);
event LotCreated(address indexed holder, uint256 lotId, uint256 quantity);
event CapitalCallCreated(uint256 indexed callId, uint256 amount);
event ContributionReceived(uint256 indexed callId, address indexed member);
event DistributionExecuted(uint256 indexed distId, uint256 totalAmount);
event InvestmentCreated(uint256 indexed investmentId, address target);
event ValuationUpdated(uint256 indexed investmentId, uint256 newValue);
```

Listing 2: Core protocol events

4.2 The Subgraph as Ledger

The protocol's subgraph maintains materialized views of:

- **Activity Feed**: Chronological record of all transactions
- **Balance History**: Point-in-time balances for any account
- **NAV History**: Daily snapshots of fund valuations
- **Capital Account**: Partner-level capital account statements
- **Tax Lots**: Cost basis and holding periods for all positions

This eliminates the need for separate accounting systems. The blockchain *is* the ledger.

4.3 Reconciliation

Traditional fund administration requires reconciliation between:

- Custodian records
- Transfer agent records
- General ledger
- Investor statements

In CapSign, these are all derived from the same source of truth—the on-chain state. Reconciliation is not a periodic batch process; it is continuous and automatic.

5 Real-Time Settlement

5.1 Atomic Execution

All CapSign transactions execute atomically within a single Ethereum transaction. A capital call, for example:

1. Verifies member eligibility
2. Transfers payment tokens from member to fund
3. Mints LP tokens to member
4. Updates capital account records
5. Emits indexable events

If any step fails, the entire transaction reverts. There is no partial state.

5.2 Settlement Finality

Once a transaction is included in a finalized block, settlement is complete. On Ethereum L2s like Base, this occurs within seconds. There is no:

- Clearing house
- Settlement window
- Counterparty risk
- Failed settlement

The implications for capital efficiency are profound. Funds can deploy capital immediately upon receipt. Investors can verify their holdings in real-time.

5.3 Comparison with Traditional Markets

Market	Settlement Time	Counterparty Risk
US Equities	T+1	Yes
Private Equity	T+30 to T+90	Yes
Real Estate	T+30 to T+60	Yes
CapSign Protocol	T+0 (seconds)	No

Table 3: Settlement comparison across market types

6 Compliance Architecture

6.1 Modular Compliance

Token transfers are subject to a configurable compliance pipeline:

$$\text{canTransfer}(from, to, amount) = \bigwedge_{m \in \text{Modules}} m.\text{check}(from, to, amount)$$

Standard modules include:

- **AccreditedInvestorModule**: Verifies accreditation status
- **KYCModule**: Ensures KYC/AML compliance
- **MaxHoldersModule**: Enforces shareholder limits (e.g., 99 for 3(c)(1))
- **LockupModule**: Enforces holding periods
- **TransferRestrictionsModule**: Issuer approval for transfers
- **ROFRModule**: Right of first refusal for secondary sales

6.2 Attestation-Based Identity

Rather than storing PII on-chain, CapSign uses *attestations*—cryptographic proofs that a trusted party has verified a claim:

```
struct Attestation {
    bytes32 schemaId;       // e.g., "accredited-investor"
    address subject;        // The verified address
    address attester;       // The verifying party
    uint64 expirationTime; // Validity period
    bytes data;             // Encoded claim data
}
```

Listing 3: Attestation structure

This enables compliance verification without exposing sensitive data on-chain.

7 Primary Capital Raises

The protocol provides comprehensive infrastructure for primary issuance—the initial sale of securities from issuer to investor.

7.1 Offering Lifecycle

An offering progresses through defined states:

1. **Draft**: Issuer configures terms, compliance requirements, and documentation
2. **Open**: Offering accepts subscriptions from qualified investors
3. **Closed**: Subscription period ends, final allocations determined
4. **Settled**: Tokens issued, funds transferred, offering complete

7.2 Offering Configuration

Each offering specifies:

- **Token**: The security being offered (equity, debt, LP units)
- **Price**: Per-unit price or pricing formula
- **Allocation**: Minimum/maximum investment amounts
- **Compliance Modules**: Required investor qualifications
- **Documents**: Subscription agreements, PPMs, side letters

7.3 Subscription Flow

The subscription flow differs based on offering type:

7.3.1 Direct Offerings (Company Equity, Debt)

1. Investor discovers offering and reviews materials
2. Investor completes required attestations (KYC, accreditation)
3. Investor signs subscription documents on-chain
4. Investor transfers payment (stablecoin or fiat bridge)
5. Issuer reviews and accepts subscription
6. Tokens are minted directly to investor's wallet
7. Lot record created with cost basis and acquisition date

7.3.2 Fund Offerings (LP Units)

Funds can operate in two modes depending on their investment strategy:

Commitment-Then-Funding Mode. Traditional PE/VC funds use the industry-standard commitment model:

1. Investor signs subscription agreement, committing to invest (e.g., $1M)
2. GP accepts subscription—investor is now an LP with a **commitment**, not tokens
3. Commitment is recorded on-chain (but no tokens minted yet)
4. Over time, GP issues capital calls for portions of committed capital
5. As LP funds each capital call, tokens are minted at current NAV per token
6. For the initial call, 1 token = $1; subsequent calls mint at prevailing NAV

Why this matters for compliance: Under Section 3(c)(1), beneficial ownership is determined by who *actually holds* fund interests—not who has promised to invest. By minting tokens only upon funding, an LP who defaults on a capital call was never a beneficial owner. The fund can simply reallocate that commitment without secondary market mechanics.

Direct Funding Mode. Funds with immediate deployment strategies—such as DeFi yield funds that deploy capital to protocols like Morpho, Aave, or Compound—can operate like direct offerings:

1. Investor subscribes with payment
2. Fund wallet receives capital and deploys it in the same transaction (or block)
3. LP tokens are minted immediately to the investor
4. NAV per token reflects the fund's on-chain positions in real-time

This mode eliminates the commitment/call cycle entirely, enabling instant liquidity deployment. It's particularly suited for tokenized yield strategies where capital is always fully deployed and NAV is continuously computable from on-chain state.
The entire flow executes on-chain with cryptographic signatures, eliminating wet signatures, wire transfers, and manual reconciliation.

7.4 Regulation Compliance

The protocol supports multiple exemption frameworks:

Exemption	**Key Requirements**	**Protocol Support**
Reg D 506(b)	Accredited + 35 sophisticated	`AccreditedInvestorModule`
Reg D 506(c)	Verified accredited only	`VerifiedAccreditedModule`
Reg S	Non-US persons	`GeographyModule`
Reg A+	$75M limit, qualified	`TieredOfferingModule`
Reg CF	$5M limit, any investor	`CrowdfundingModule`
3(c)(1)	100 beneficial owners	`MaxHoldersModule`
3(c)(7)	Qualified purchasers only	`QualifiedPurchaserModule`

Table 4: Regulatory exemption support

7.5 Document Integration

Offering documents are registered on-chain with content-addressed storage:

$$\text{DocumentId} = \text{keccak256}(\text{content})$$

This ensures:

- Immutable record of what was disclosed
- Cryptographic proof of investor acknowledgment
- Tamper-evident audit trail
- Integration with e-signature workflows

7.6 Payment Rails

The protocol supports multiple payment methods:

- **Stablecoin**: Direct USDC/USDT transfer (instant settlement)
- **Fiat Bridge**: ACH/wire via integrated banking partner
- **Crypto**: ETH or other tokens with on-chain conversion

All payment methods result in the same on-chain outcome: tokens in the investor's wallet, funds in the issuer's wallet.

8 Capital Operations

8.1 Capital Calls

Capital calls are the mechanism by which GPs draw down committed capital from LPs:

1. **Call Creation**: GP creates a capital call specifying amount per unit, due date, and purpose
2. **Obligation Calculation**: System calculates each LP's pro-rata obligation based on their commitment
3. **Notification**: LPs receive on-chain notification of amount due
4. **Contribution**: LP transfers the called amount (stablecoin or fiat bridge)
5. **Token Minting**: Upon receipt of funds, LP tokens are minted to the contributor
6. **Default Handling**: If LP fails to fund, commitment is marked as defaulted

8.1.1 Token Economics

Tokens are minted at the current NAV per token at the time of each capital call:

$$\text{Tokens Minted} = \frac{\text{Capital Contributed}}{\text{NAV per Token}}$$

For the initial capital call (before any fund activity), NAV per token is set to \$1.00 by convention. Subsequent capital calls mint tokens at the prevailing NAV:

$$\text{NAV per Token} = \frac{\text{Total Fund Assets} - \text{Liabilities} - \text{Accrued Fees}}{\text{Total Tokens Outstanding}}$$

This ensures all LPs—whether they funded in the first call or a later one—receive tokens proportional to their economic contribution relative to the fund's current value. An LP joining via a later capital call when the fund has appreciated receives fewer tokens per dollar, reflecting that existing LPs created that appreciation.

8.1.2 Commitment Lifecycle

Commitments are tracked separately from token ownership:

Status	**Description**	**Token Impact**
ACTIVE	In good standing	Tokens minted as funded
DEFAULTED	Failed capital call	No further tokens; may forfeit
CANCELLED	GP cancelled pre-funding	None (never a beneficial owner)
WITHDRAWN	LP exited (if permitted)	Tokens redeemed

Table 5: Commitment status lifecycle

8.2 Distributions

Distributions flow capital from the fund back to LPs according to the fund's waterfall structure. The protocol implements a hybrid off-chain/on-chain architecture that combines the computational flexibility of off-chain calculation with the verifiability and atomicity of on-chain execution.

8.2.1 Waterfall Calculation

Distribution waterfalls—the rules governing how proceeds are allocated among LPs, GP, and carried interest—can be arbitrarily complex: preferred returns, catch-up provisions, tiered carry, clawback reserves, and LP-specific side letter terms. Rather than encoding this complexity in smart contracts

(which would be gas-prohibitive and inflexible), the protocol calculates waterfalls off-chain using the subgraph as its data source.
The subgraph maintains complete capital account history for each LP:

- Cumulative contributions (funded capital calls)
- Cumulative distributions received
- Unreturned capital balance
- Preferred return accrual
- Current profit/loss position

The application layer queries this data, applies the fund's waterfall logic (which may include custom terms per LP), and computes the precise distribution amount for each member.

8.2.2 Atomic Execution via Batched UserOps

Once the waterfall is calculated, the fund manager reviews and approves the distribution. The application then constructs a batch of ERC-4337 UserOperations—one transfer per recipient—bundled into a single transaction from the fund wallet:

```
// Fund wallet executes batch of transfers as single atomic tx
FundWallet.executeBatch([
    { to: LP1, value: 0, data: transfer(LP1, 50000 USDC) },
    { to: LP2, value: 0, data: transfer(LP2, 30000 USDC) },
    { to: LP3, value: 0, data: transfer(LP3, 20000 USDC) },
    { to: GP,  value: 0, data: transfer(GP, 10000 USDC) },  // Carry
]);
```

Listing 4: Batched distribution execution

This architecture provides:

- **Atomicity**: All distributions succeed or all revert—no partial distributions
- **Verifiability**: The on-chain transaction is the auditable record; anyone can verify amounts match the waterfall
- **Efficiency**: Single transaction regardless of LP count, minimizing gas overhead
- **Flexibility**: Waterfall logic can be updated without contract changes

Distribution events are indexed by the subgraph, updating each LP's capital account and generating the accounting records needed for K-1 preparation.

8.3 Fund Fees

The protocol supports on-chain fee accrual and settlement. The mechanics described here reflect the typical PE/VC model, but the protocol is agnostic to fee structure—funds can implement fees as:

- **Cash payment** (traditional): GP receives stablecoin, immediate liquidity, ownership percentage unchanged
- **Token dilution** (DeFi-style): GP receives newly minted LP tokens, increasing their ownership stake but deferring liquidity—aligning GP compensation with fund performance

The choice depends on fund structure and GP preference. Below we describe the traditional cash-based model.

8.3.1 Management Fees

Management fees (typically 1.5–2% annually on committed or invested capital) are paid in cash (stablecoin) from fund assets to the GP:

$$\text{Quarterly Fee} = \text{Fee Basis} \times \frac{\text{Annual Rate}}{4}$$

Where Fee Basis is either committed capital (during investment period) or invested capital (after investment period), per the fund's LPA. The fee payment is executed as a stablecoin transfer from the fund wallet to the GP wallet, reducing fund NAV directly.

8.3.2 Carried Interest

Carried interest (typically 20% of profits above a hurdle) is calculated at distribution time following the fund's waterfall:

1. Return LP capital contributions (return of capital)
2. Pay preferred return to LPs (if applicable)
3. GP catch-up (if applicable)
4. Split remaining profits per carry percentage

Carry is paid in cash from distribution proceeds—the GP receives their carry allocation as part of the batched distribution transaction, not through token minting.

8.3.3 NAV with Fee Accrual

For accurate real-time NAV between fee payment dates, accrued but unpaid fees are deducted:

$$\text{Net NAV} = \text{Gross Assets} - \text{Liabilities} - \text{Accrued Mgmt Fees} - \text{Accrued Expenses}$$

This ensures LP token holders always see the "after-fee" value of their position, even mid-quarter.

8.4 Secondary Transfers

The protocol supports controlled secondary liquidity:

1. Seller initiates transfer request
2. ROFR module notifies existing members of opportunity
3. Members may exercise ROFR within specified window
4. If ROFR expires unexercised, GP may approve external transfer
5. Compliance modules verify buyer eligibility
6. Transfer executes atomically

9 Tax Infrastructure

9.1 Cost Basis Tracking

The lot-level accounting model enables precise cost basis tracking:

$$\text{Gain(lot)} = \text{SaleProceeds} - \text{CostBasis(lot)}$$

The protocol supports multiple cost basis methods:

- FIFO (First In, First Out)
- LIFO (Last In, First Out)

- HIFO (Highest In, First Out)
- Specific Identification

9.2 Wash Sale Detection

For tax lots, the protocol tracks potential wash sales:

$$\text{IsWashSale}(\text{sale}, \text{purchase}) = |\text{sale.date} - \text{purchase.date}| \leq 30 \text{ days}$$

Disallowed losses are automatically added to the cost basis of replacement shares.

9.3 K-1 Generation

For partnership structures, the protocol computes Schedule K-1 allocations:

- Ordinary income
- Short-term capital gains
- Long-term capital gains
- Interest income
- Dividend income
- Section 199A deductions

These are allocated to partners based on their capital account percentages.

10 Use Cases

10.1 Fund Administration

The ownership graph transforms fund administration from a labor-intensive back-office function into an automated, real-time system. A fund's wallet owns its portfolio company tokens; LP wallets own fund tokens; the graph captures this hierarchy natively.

Capital calls become on-chain notifications with automatic obligation calculation per LP. Contributions mint tokens at current NAV. Distributions compute waterfalls off-chain and execute atomically. K-1 data is derived directly from the subgraph's capital account history—no reconciliation required.

For GPs managing multiple funds, the hierarchical structure enables consolidated reporting across vehicles while maintaining strict fund-level segregation. Co-investment SPVs are simply child nodes in the graph, inheriting compliance requirements from their parent fund.

10.2 Private Companies

Private companies use CapSign to maintain their cap table as a living on-chain record. Each shareholder's wallet holds tokens representing their equity; the company wallet is the token's issuer. Option grants, RSU vesting, and SAFE conversions execute as token operations with automatic lot creation for tax basis tracking.

The ownership graph captures complex structures naturally: an employee holds options through their personal wallet, which may be owned by their family trust, which may be owned by multiple beneficiaries. Beneficial ownership rolls up automatically. When a 409A valuation updates, NAV per token reflects the new fair market value across all holders.

Secondary transfers—common in late-stage private companies—execute through the compliance pipeline with ROFR enforcement, transfer restrictions, and automatic cap table updates.

10.3 Private Credit

Private credit represents a natural extension of the ownership graph to debt instruments. A loan is a token: the lender's wallet holds an asset; the borrower's wallet carries a corresponding liability. Interest accruals, payment schedules, and covenant tracking are token-level operations.

For fund structures that deploy capital into credit strategies—whether direct lending, asset-based financing, or structured products—the protocol enables same-block deployment: investor subscribes, fund receives capital, capital deploys to Morpho or other DeFi lending protocols, all in a single transaction. NAV reflects on-chain positions continuously.

This architecture points toward programmable commercial banking: credit facilities where drawdowns, repayments, and interest calculations execute automatically based on on-chain state. The line between "fund administration" and "banking infrastructure" blurs when both are nodes in the same ownership graph.

11 Architecture

11.1 Smart Contract Layer

The protocol is deployed on Ethereum L2 (Base) and implements the EIP-2535 diamond standard throughout. Every entity—wallets, tokens, offerings, escrows—is a diamond proxy composed of modular facets. This architecture enables contracts to evolve over time without migration, share implementation code across instances, and maintain a single address identity throughout their lifecycle.

11.1.1 Infrastructure Contracts

The protocol's foundation is the **Diamond Factory**—a generic CREATE2 factory diamond that deploys new diamonds with deterministic addresses. The factory accepts any valid facet configuration, enabling permissionless deployment of wallets, tokens, offerings, and other protocol entities.

CapSign Inc. additionally maintains a **Facet Registry**—a versioned registry of audited facet implementations used by our platform. The registry tracks facet names, versions, selectors, and deprecation status. When deploying through the CapSign interface, the factory validates cuts against this registry, ensuring users only deploy approved, audited code. This is an operational security measure for our platform; developers building directly on the protocol may use the Diamond Factory without registry validation if they prefer to manage their own facet security.

11.1.2 Domain Factories

Specialized factory diamonds provide high-level APIs for creating protocol entities:

- **Wallet Factory**: Creates wallet diamonds for entities with configurable facet sets (core, signature, documents, identity, paymaster policy)
- **Token Factory**: Creates security token diamonds with ERC-20, compliance, lot tracking, and distribution facets
- **Offering Factory**: Creates offering diamonds for primary issuance with compliance modules, document management, and payment handling
- **Escrow Factory**: Creates escrow diamonds for holding funds during transactions with configurable release conditions

Each factory validates inputs, configures appropriate facets, and initializes the diamond in a single atomic transaction.

11.1.3 Upgradeability

Because every entity is a diamond, wallets, tokens, and offerings can be upgraded to incorporate new facets or replace existing ones. However, upgradeability follows the ownership hierarchy:

- **CapSign Inc. publishes new facet versions** to the Facet Registry. These may include bug fixes, new features, or gas optimizations.
- **Entity owners decide whether to upgrade.** CapSign Inc. cannot unilaterally modify deployed diamonds. Only the entity's authorized signers can execute a diamond cut to add, replace, or remove facets.
- **Upgrades are opt-in and auditable.** Each upgrade is an on-chain transaction signed by the entity, creating a permanent record. Entities can remain on older facet versions indefinitely if they prefer stability over new features.

This model balances the benefits of upgradeable infrastructure (bug fixes, feature additions) with entity sovereignty—no one can modify your contracts without your explicit authorization.

11.1.4 Hierarchical Access Control

Access control follows the ownership hierarchy. Every diamond includes an `AccessControlFacet` providing local role-based permissions using efficient bit-flags (supporting 256 roles per user). Critical to the architecture is the *authority delegation* pattern:

1. When an issuer wallet creates a token, the token's `authority` is set to the issuer wallet address
2. The issuer wallet includes a `WalletAccessManagerFacet` implementing OpenZeppelin's `IAccessManager` interface
3. When protected functions on the token are called, the token first checks with its authority (the wallet) whether the caller is permitted
4. The wallet checks if the caller has the appropriate role on the wallet itself

This pattern means permissions flow through the ownership graph: an entity's wallet controls its tokens and offerings, and the entity's authorized signers can operate those instruments according to their assigned roles. There is no global "protocol-level" access manager—each wallet manages its own permissions for its owned assets.

```
// Token checks its authority (issuer wallet) for permissions
function transfer(address to, uint256 amount) external {
    if (!canCall(msg.sender, this.transfer.selector)) revert Unauthorized();
    _transfer(msg.sender, to, amount);
}

function canCall(address caller, bytes4 selector) internal view returns (bool) {
    // First check external authority (issuer wallet)
    if (authority != address(0)) {
        (bool allowed,) = IAccessManager(authority).canCall(
            caller, address(this), selector);
        if (allowed) return true;
    }
    // Fall back to local role checks
    return hasRole(caller, requiredRole[selector]);
}
```

Listing 5: Authority delegation for access control

11.1.5 Identity and Compliance Verification

The protocol supports two complementary modes for investor identity verification, selectable per offering based on issuer preference:

Direct Verification Mode. Issuers or their designated KYC providers record verification status directly on the offering diamond via the `ComplianceAdminFacet`. This stores KYC status and investor classifications (accredited, qualified purchaser, sophisticated, etc.) in the offering's on-chain storage. The issuer can designate trusted third-party KYC providers who are authorized to set verification status on their behalf.

```
// Issuer or trusted provider sets investor KYC status
ComplianceAdminFacet(offering).setKYCStatus(
    investor,       // Investor wallet address
    true,           // Verified
    expirationTime  // Optional expiration (0 = never)
);

// Set investor classification (accredited, qualified purchaser, etc.)
ComplianceAdminFacet(offering).setClassificationStatus(
    investor,
    [keccak256("ACCREDITED"), keccak256("QUALIFIED_PURCHASER")],
    expirationTime
);
```

Listing 6: Direct KYC verification by issuer

This mode is simpler, faster (direct storage access), and gives issuers full control over who can invest. The tradeoff is that verification is offering-specific—investors must be verified separately for each offering.

Portable Attestation Mode. Investors store Ethereum Attestation Service (EAS) attestation UIDs on their wallet diamonds via the `WalletIdentityFacet`. These attestations are portable: an investor verified once can present the same attestation across multiple offerings. Compliance modules query the investor's wallet for relevant attestations during subscription.
This mode reduces friction for repeat investors and enables third-party attestation providers to issue credentials once that work protocol-wide. The tradeoff is additional complexity and reliance on external attestation infrastructure.

Hybrid Approach. Compliance modules support both modes simultaneously. During compliance checks, modules first query the offering's direct storage (cheapest path), then fall back to checking wallet attestations if provided. This allows issuers to use whichever mode fits their workflow while investors benefit from portable credentials when available.

11.2 Indexing Layer

A Graph Protocol subgraph indexes all protocol events, maintaining materialized views that would be prohibitively expensive to compute on-chain:

- **Activity Feed**: Chronological record of all protocol actions with rich metadata
- **Balance Snapshots**: Point-in-time balances for any account, enabling historical queries
- **NAV History**: Daily fund valuations computed from on-chain asset positions and price feeds
- **Capital Accounts**: Partner-level capital account statements with contribution/distribution history
- **Tax Lot History**: Complete audit trail of lot creation, transfers, and dispositions with cost basis

The subgraph transforms raw blockchain events into queryable accounting records, eliminating the need for separate off-chain databases while maintaining the blockchain as the authoritative source of truth.

11.3 Application Layer

CapSign provides a web application serving as the primary interface to the protocol:

- **Entity Onboarding**: KYC/AML verification, accreditation checks, entity structuring
- **Wallet Management**: Passkey-based authentication, signer management, role assignment
- **Token Operations**: Issuance, transfers, corporate actions, cap table visualization
- **Fund Administration**: Capital calls, distributions, NAV calculations, investor statements
- **Document Workflow**: Template generation, e-signature collection, on-chain registration
- **Investor Portal**: Holdings dashboard, document access, subscription management

The application uses account abstraction (ERC-4337) for gasless transactions, enabling traditional user experiences without requiring users to hold ETH or understand blockchain mechanics. Gas sponsorship operates at two levels:

- **Protocol Paymaster**: CapSign Inc. operates a paymaster that sponsors transactions for platform users, abstracting away gas costs entirely for standard operations.
- **Wallet Paymaster Policies**: Each wallet can configure its own sponsorship policy, enabling entities to sponsor gas for their investors, employees, or related wallets. For example, a fund wallet can sponsor all transactions from its LPs, or a corporate wallet can sponsor its subsidiaries. This creates a gas sponsorship hierarchy that mirrors the ownership graph.

12 Security Considerations

12.1 Smart Contract Security

All protocol contracts undergo:

- Formal verification of critical invariants
- Multiple independent security audits
- Bug bounty programs
- Gradual rollout with upgrade timelock

12.2 Access Control

The protocol implements defense-in-depth:

- Role-based access control (RBAC)
- Multi-signature requirements for sensitive operations
- Time-delayed execution for administrative changes
- Account abstraction for improved key management

12.3 Privacy

Sensitive data handling:

- PII stored off-chain with encrypted references
- Attestations prove claims without revealing data
- Zero-knowledge proofs for selective disclosure (roadmap)

13 Governance

13.1 Protocol vs. Platform

A critical distinction: the *protocol* is code—permissionless smart contracts deployed on a public blockchain. Code cannot be regulated; it simply executes according to its logic. Anyone can interact with the deployed contracts directly, build alternative interfaces, or fork the codebase.

CapSign Inc. is a business entity that operates a platform—an interface to the protocol. The company is subject to applicable regulations, maintains compliance programs, and can be held accountable. When institutional investors require a responsible counterparty, they contract with CapSign Inc., not with the protocol.

This separation is fundamental: the protocol provides infrastructure; the platform provides service. Users who prefer self-custody and direct contract interaction can use the protocol without CapSign Inc. Users who want a managed experience with support, compliance, and accountability use the platform.

13.2 Protocol Governance

The protocol has parameters that require ongoing stewardship:

- **Facet Registry**: Which facet implementations are approved for deployment through the CapSign platform. New facets require security audits before registration.
- **Factory Configuration**: Creation fees on token and offering factories, treasury addresses, and payment discounts.
- **Paymaster Policies**: Which operations the protocol paymaster sponsors, rate limits, and abuse prevention.

Note on fees: As of this writing, no protocol fees are enabled—deploying tokens and offerings through the factories is free. The factory contracts include fee parameters that can be activated in the future, but there is no current timeline for doing so.

13.2.1 Current Model

CapSign, Inc. currently serves as protocol steward with the following responsibilities:

- Maintaining and auditing smart contract code
- Operating infrastructure (subgraph indexers, bundlers, paymasters)
- Responding to security incidents
- Coordinating with regulators and legal counsel
- Making protocol upgrade decisions

All protocol contracts are deployed with transparent, verifiable code. Users retain self-custody of their assets and can interact with deployed contracts directly—the protocol is permissionless at the contract layer even while the CapSign platform provides a curated experience.

13.2.2 Future Considerations

As the ecosystem matures, protocol stewardship may evolve to include input from major stakeholders (fund administrators, institutional users, infrastructure operators). The protocol layer could eventually operate with minimal ongoing governance—immutable contracts that simply work. Platform-level governance (CapSign Inc.'s policies, compliance programs, service terms) will continue to evolve with regulatory requirements and customer needs, but that is distinct from protocol governance.

13.3 Entity Governance

While protocol governance is centralized, *entity* governance is fully configurable by each organization using the protocol:

- **Signing Thresholds**: Multi-signature requirements for different operation types (e.g., 2-of-3 for transfers, 3-of-5 for distributions)
- **Role-Based Permissions**: Granular roles (admin, operator, viewer) with function-level access control. Signers can be scoped to specific operations.
- **Signer Types**: Support for both EOA signers and passkey-based authentication, enabling secure access without seed phrase management.
- **Document-Based Resolutions**: On-chain registration of board resolutions, member votes, and governance documents with cryptographic proof of acknowledgment.
- **Delegation**: Authorized signers can be granted limited roles (e.g., a fund accountant who can view but not transfer, or a compliance officer who can approve KYC but not execute trades).

13.3.1 On-Chain Voting and Execution

For entities requiring formal member voting, the protocol integrates with the OpenZeppelin Governor pattern. Token holders can propose actions, vote on-chain, and—upon reaching quorum and approval—execute the outcome directly through the entity wallet. This enables:

- **Binding Votes**: Voting results are not merely advisory; approved proposals execute automatically via the entity wallet
- **Transparent Process**: Proposal creation, voting, and execution are fully on-chain and auditable
- **Configurable Parameters**: Quorum thresholds, voting periods, proposal thresholds, and timelocks are set per entity
- **Token-Weighted Voting**: Voting power derived from token ownership, with support for delegation

This is particularly relevant for funds structured as DAOs, SPVs with many members, or any entity where material decisions require formal member approval with verifiable outcomes.
This separation—centralized protocol governance with decentralized entity governance—reflects the reality that protocols need accountable stewards while organizations need flexible, self-sovereign control over their own operations.

14 Conclusion

CapSign Protocol represents a fundamental reimagining of private capital markets infrastructure. By modeling ownership as a hierarchical graph, maintaining full balance sheet representation, enabling automatic ledgering, and providing real-time settlement, the protocol eliminates entire categories of operational overhead while increasing transparency and reducing risk.
The implications extend beyond operational efficiency. When ownership is unambiguous, settlement is instant, and records are immutable, new possibilities emerge: frictionless secondary markets, real-time NAV lending, automated compliance, and global access to private market opportunities.
We believe this architecture will become the standard for representing ownership in the digital age.

Acknowledgments

The CapSign team thanks Nick Mudge, creator of EIP-2535 (the diamond standard), whose modular proxy architecture forms the foundation of our smart contract design. We thank our angel investors for their early belief in this vision. We also thank the Ethereum community, the Base team at Coinbase, the ERC-4337 account abstraction contributors, and the broader ecosystem for the infrastructure upon which this protocol is built.

References

[1] Nick Mudge, *EIP-2535: Diamonds, Multi-Facet Proxy*, Ethereum Improvement Proposals, 2020.
https://eips.ethereum.org/EIPS/eip-2535

[2] Vitalik Buterin, Yoav Weiss, Dror Tirosh, et al., *ERC-4337: Account Abstraction Using Alt Mempool*, Ethereum Improvement Proposals, 2021.
https://eips.ethereum.org/EIPS/eip-4337

[3] Ethereum Attestation Service, *EAS: A Standard for Ethereum Attestations*, 2023.
https://attest.org

[4] Coinbase, *Base: Ethereum L2*, 2023.
https://base.org