The Tokenized Deposit: Real-Time Settlement Infrastructure for the Digital Economy

October 2025

Aurélien Bonnel
aurelien.bonnel@n3xt.io

Andrew McKenzie
andrew.mckenzie@n3xt.io

Kyle O'Donnell
kyle.odonnell@n3xt.io

Tiffiney Peterson
tiffiney.peterson@n3xt.io

Jeffrey Wallis
jeffrey.wallis@n3xt.io

Scott Shay
scott.shay@n3xt.io

Abstract

This paper describes the Tokenized Deposit: a fully-reserved, bank-issued representation of a deposit that serves as a method of transfer of deposits and as foundational settlement infrastructure. Issued by a chartered Special Purpose Depository Institution (SPDI), the Tokenized Deposit is the legal and accounting equivalent of a U.S. Dollar deposit at the bank, represented as a token on public blockchains. It is the same product the institution offers on its private blockchain, now represented on public blockchains—no digital asset is created. The system eliminates the fundamental limitations of legacy bank rails through a novel architecture that combines public blockchain settlement with real-time, event-driven reconciliation. By maintaining continuous synchronization between on-chain deposit token balances and off-chain bank reserves through a "Digital Lockbox" mechanism, the system delivers cryptographic proof of full reserves while providing the speed, programmability, and transparency of a blockchain-native representation of deposits. This architecture eliminates counterparty risk, settlement ambiguity, and the accounting complexities of volatile crypto-assets, enabling a new era of secure, efficient, and compliant transfer of deposits.

1 Introduction

1.1 The Dual Challenge: Transfer of Deposits and Reconciliation

The global financial system operates on legacy infrastructure fundamentally incompatible with the speed, openness, and programmability of the internet. Cross-border transfer of deposits remains slow, opaque, and expensive. Treasury operations are fragmented across intermediaries and incompatible bank rails. These structural inefficiencies create significant economic costs and introduce systemic counterparty and operational risks.

Beyond the bank rails themselves, traditional financial institutions face an equally critical challenge in reconciliation—the process of ensuring that multiple systems accurately reflect the same financial reality. Legacy batch processing systems, operating on periodic reconciliation cycles that may run hours or even days after transactions occur, create windows of uncertainty where account balances may not reflect actual positions. This temporal gap introduces operational risk, complicates fraud detection, and creates regulatory compliance challenges.

The Tokenized Deposit addresses both challenges simultaneously. It is a true reflection of a bank deposit—not an approximation—built on full reserves and maintained through real-time reconciliation that provides continuous cryptographic proof of the 1:1 reserve ratio. It is the same product the institution offers on its private blockchain, now represented on public blockchains; the institution is not creating a new digital asset. This synthesis of blockchain settlement and event-driven financial infrastructure eliminates the trade-offs that have constrained tokenized deposit adoption.

2 The Solution: A True Tokenized Deposit with Real-Time Reserve Proof

2.1 The Foundation: Full-Reserve Narrow Banking

We offer a tokenized representation of a U.S. Dollar deposit at the bank—the same deposit product we offer on our private blockchain, now represented on public blockchains.

As a regulated **Special Purpose Depository Institution (SPDI)**, we operate as a narrow bank on a full-reserve basis. We do not engage in lending, credit, or other speculative activities. All customer deposits are held in cash and short-term U.S. Treasury bills, fully segregated, auditable, and protected. Our Tokenized Deposit is a direct, 1:1 representation of a U.S. Dollar deposit held in custody at our bank.

2.2 The Innovation: Real-Time Reconciliation Architecture

What distinguishes the Tokenized Deposit from other bank-issued digital tokens is its real-time reconciliation architecture. Rather than relying on periodic attestations or batch processing to verify reserves, the system maintains continuous synchronization between on-chain deposit token supply and off-chain bank reserves through an event-driven architecture.

Every transaction—whether a mint, burn, or transfer—triggers an immediate cascade of events that update multiple systems in near real-time. A blockchain monitoring service captures on-chain events as they achieve finality, publishes them to an event processing infrastructure, and triggers corresponding updates to the Core Ledger, the definitive source of truth for customer balances and reserve allocations.

This architecture provides several critical advantages:

- **Continuous Reserve Verification:** At any moment, the total on-chain deposit token supply can be cryptographically verified against the total value locked in customer Digital Lockboxes on the Core Ledger, providing transparent proof of full reserves.
- **Immediate Inconsistency Detection:** Any discrepancy between on-chain and off-chain state is detected within seconds rather than hours or days, enabling rapid investigation and resolution.
- **Elimination of Settlement Windows:** Traditional batch systems create temporal gaps where transactions are "in flight" between systems. Real-time reconciliation eliminates these windows, ensuring that settlement is truly final when blockchain finality is achieved.
- **Enhanced Operational Security:** Continuous monitoring enables immediate detection of unauthorized minting, burning, or other anomalous activities, providing a critical security layer beyond smart contract access controls.

3 System Architecture and Protocol Specification

The system is a hybrid model designed to merge the security and compliance of a regulated financial institution with the utility and accessibility of public blockchains. This is achieved through three integrated layers: a blockchain-native compliance framework, real-time event processing infrastructure, and a cryptographically-verifiable ledger system.

3.1 Core Architectural Components

The system architecture comprises three distinct but interconnected layers, each serving a specific purpose in maintaining the integrity and auditability of the Tokenized Deposit:

3.1.1 Layer 1: The Blockchain (Source of Truth for Token State)

The public blockchain serves as the immutable, authoritative record of all deposit balances and transactions. Two smart contracts operate on this layer:

- **Tokenized Deposit Contract (ERC-20):** Manages token balances and enforces transfer rules
- **Identity NFT Contract (ERC-721):** Maintains the whitelist of authorized participants

The blockchain's role is singular and critical: it is the definitive source of truth for *who holds how many deposit tokens at any given moment.* All token state—balances, supply, transaction history—is stored here, immutably and transparently.

3.1.2 Layer 2: The Accounting Ledger (Source of Truth for Reserve Allocation)

The Accounting Ledger is the bank's internal financial system that maintains the definitive record of customer deposits, account balances, and reserve allocations. This is a private blockchain deployment, mirrored by a traditional ACID-compliant database system that tracks:

- **Customer Accounts:** General deposit balances (traditional bank accounts)
- **Digital Lockboxes:** Ring-fenced reserves backing on-chain deposit tokens
- **Cash-in-Transit Accounts:** Temporary holding for in-flight deposits/withdrawals

- **FBO (For Benefit Of) Accounts:** Omnibus accounts securing client funds

The **Digital Lockbox** is an accounting construct within this ledger. When a customer requests tokenization, funds are transferred from their general account balance into their Digital Lockbox, where they become segregated reserves backing on-chain deposit tokens. These funds cannot be used for any other purpose—they are ringfenced until the corresponding tokens are burned.

The Accounting Ledger also maintains a **General Ledger** for double-entry bookkeeping, tracking all movements with proper debits and credits across account categories (Deposits, Tokenized Deposits, Cash-in-Transit, etc.).

Critically: *The Accounting Ledger determines which customer owns which reserves, but the blockchain determines how many tokens exist and who holds them.* These two systems must remain perfectly synchronized.

3.1.3 Layer 3: The Real-Time Reconciliation Infrastructure (The Synchronization Layer)

The reconciliation infrastructure is the critical innovation that maintains continuous synchronization between the blockchain and the Accounting Ledger. This event-driven architecture ensures that:

$$\sum_{\text{all customers}} \text{Digital Lockbox Balance} = \text{Total on-chain deposit token Supply} \tag{1}$$

This equality is maintained continuously and verified in real-time, providing cryptographic proof of full reserves at every moment.

3.2 The Real-Time Reconciliation Infrastructure: Detailed Architecture

The reconciliation infrastructure is the critical innovation that bridges the blockchain and the Accounting Ledger. This event-driven architecture processes every blockchain state change in real-time, ensuring continuous synchronization and immediate discrepancy detection.

3.2.1 Reconciliation Components

Block-Listener Service A fault-tolerant, distributed service that continuously monitors the blockchain for Tokenized Deposit smart contract events. The Block-Listener:

- Subscribes to blockchain nodes to receive real-time event notifications
- Detects three critical event types: `Transfer` (includes mints from zero address and burns to zero address), `Approval`, and `Admin` events
- Waits for transaction finality (e.g., 12 block confirmations on Ethereum or probabilistic finality on other chains) before considering events as settled
- Extracts event details: sender, receiver, amount, transaction hash, block number, log index
- Publishes finalized events to the Event Broker with full blockchain provenance

This service never processes unfinalized transactions, eliminating the risk of reconciling events that could be reversed due to chain reorganizations.

Event Broker A high-throughput, durable message queue (e.g., Apache Kafka, AWS Kinesis, GCP PubSub...) that serves as the central nervous system of the reconciliation architecture. The Event Broker:

- Receives events from the Block-Listener and other event producers
- Maintains strict ordering guarantees based on blockchain transaction order (block number, transaction index, log index)
- Provides message durability by persisting all events to disk
- Enables multiple downstream customers to process the same event stream independently
- Implements retry mechanisms with exponential backoff for transient failures
- Routes failed events to dead letter queues for manual investigation

Reconciliation Service The core processing engine that maintains synchronization between blockchain token state and Accounting Ledger reserve allocation. For each blockchain event, the Reconciliation Service:

- Consumes the event from the Event Broker
- Determines the event type (mint, burn, transfer)
- Executes the corresponding Accounting Ledger operation:

  - **Mint Event:** Marks the Digital Lockbox funds as "active" and available for on-chain use
  - **Burn Event:** Releases funds from Digital Lockbox back to customer's general account
  - **Transfer Event:** Transfers Digital Lockbox allocation from sender to receiver
- Updates the General Ledger with appropriate debit/credit entries
- Logs the reconciliation action with full audit trail (event ID, timestamp, blockchain transaction hash, account changes)
- Performs continuous verification: compares sum of all Digital Lockboxes against on-chain total supply
- Triggers alerts if any discrepancy is detected (expected: zero discrepancy at all times)

Discrepancy Detection Service A dedicated monitoring service that continuously verifies the fundamental invariant:

$$\sum_{\text{all customers}} \text{Digital Lockbox}_i = \text{totalSupply}() \tag{2}$$

Where `totalSupply()` is queried directly from the blockchain smart contract. This service:

- Runs verification checks every 60 seconds (configurable)
- Queries the blockchain for current total supply
- Queries the Accounting Ledger for sum of all Digital Lockboxes
- Compares the two values with zero tolerance
- If discrepancy detected: immediately halts minting operations, triggers emergency alerts, initiates forensic investigation
- Maintains historical log of all verification checks for audit purposes

Minter and Burner Services Secure, access-controlled services that are the *only* components authorized to invoke mint/burn operations on the blockchain:

- **Minter Service:** Receives mint instructions from the Accounting Ledger (after funds are locked in Digital Lockbox), verifies authorization, executes on-chain mint transaction with multi-signature requirements
- **Burner Service:** Receives burn instructions from the Accounting Ledger (after customer withdrawal request), verifies authorization, executes on-chain burn transaction
- Both services operate with hardware security modules (HSMs) for private key management
- All operations are logged and require multi-party authorization above certain thresholds

NFT Admin Service A secure service that manages Identity NFT lifecycle:

- Mints Identity NFTs to customer wallets upon successful KYC/KYB completion
- Burns Identity NFTs when customers are off-boarded or sanctioned
- Maintains mapping between customer legal identity, wallet address, and NFT token ID
- Operates with strict access controls and comprehensive audit logging

Event Store An append-only, immutable data store that maintains a complete historical record:

- Stores every event processed: blockchain events, reconciliation actions, verification results
- Enables point-in-time state reconstruction for any past moment
- Supports regulatory compliance reporting and forensic investigations
- Facilitates disaster recovery through event replay capability
- Provides data for analytics, monitoring, and system optimization

3.3 Event Ordering and Consistency Guarantees

Maintaining proper event ordering is critical for financial system integrity. The architecture implements sophisticated ordering mechanisms that ensure events are processed in the correct sequence, preventing race conditions and maintaining data consistency across distributed systems.

The system employs **blockchain-based ordering** where events are ordered using blockchain transaction sequences, incorporating block number, transaction index, and log index to create a deterministic ordering that reflects the actual sequence of financial activities. This approach ensures that the order of events in the reconciliation system matches the order of transactions on the blockchain, maintaining consistency between on-chain and off-chain records.

Idempotency mechanisms handle duplicate events gracefully without causing inconsistencies. Each event contains a unique identifier (derived from blockchain transaction hash and log index) that allows the Reconciliation Service to detect and skip duplicate events, ensuring that if the same event is processed multiple times, the system state remains unchanged.

The architecture provides **resume capability**, allowing systems to resume processing from the last successfully processed event after system restarts or failures. This ensures that no events are lost or processed out of order, even during system maintenance or unexpected outages.

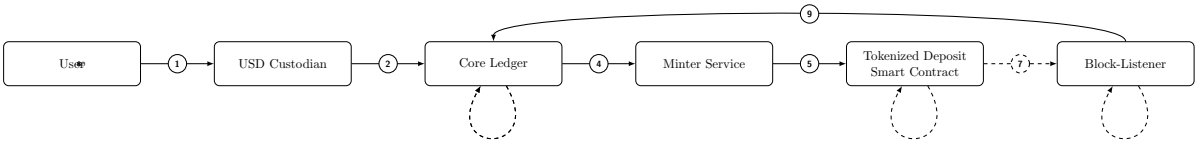

Figure 1: Deposit and Minting Flow with Real-Time Reconciliation. **Legend:** (1) Initiate Fiat Transfer; (2) Notify Core Ledger of Incoming Funds; (3) Credit User & Lock Funds in Digital Lockbox; (4) Instruct Minter to Mint; (5) Call `mint()` on Smart Contract; (6) Increase on-chain deposit token Balance; (7) Emit "TokensMinted" Event; (8) Block-Listener Waits for Finality; (9) Notify Core Ledger Mint is Final; (10) Mark Digital Lockbox Funds as Available for On-Chain Use.

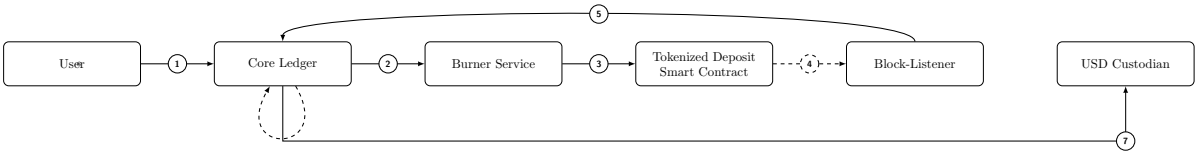

Figure 2: Withdrawal and Burning Flow with Real-Time Reconciliation. **Legend:** (1) User Requests Withdrawal; (2) Core Ledger Instructs Burn; (3) Call `burnFrom()` on Smart Contract; (4) Emit "TokensBurned" Event; (5) Block-Listener Notifies Core Ledger Burn is Final; (6) Release Funds from Digital Lockbox; (7) USD Custodian Initiates Wire Payout.

4 Complete Transaction Lifecycle with Real-Time Reconciliation

This section details the complete end-to-end flows, emphasizing the clear separation between blockchain token state and Accounting Ledger reserve allocation, and how real-time reconciliation maintains their synchronization.

4.1 Deposit and Minting Flow

The deposit process demonstrates the complete lifecycle from fiat receipt to on-chain deposit token availability, with continuous reconciliation ensuring reserve backing (see Figure 1):

Phase 1: Fiat Receipt and Reserve Allocation (Accounting Ledger Operations)

1. **Fiat Transfer:** Customer initiates a wire transfer or ACH deposit to the institution's FBO (For Benefit Of) account through the traditional bank rails.

2. **Custodian Notification:** The USD Custodian notifies the Accounting Ledger system of received funds.

3. **Accounting Ledger Update:** The Accounting Ledger executes several operations atomically:

   - Credits the customer's general deposit account
   - General Ledger entries: Debit "FBO Bank Account", Credit "Customer Deposits"
   - Logs the deposit with full audit trail

4. **Tokenization Request:** Customer requests to tokenize a specific amount (e.g., $100) through the customer interface.

5. **Digital Lockbox Allocation:** The Accounting Ledger transfers $100 from the customer's general account to their Digital Lockbox:

    - Internal account transfer: General Account → Digital Lockbox
    - General Ledger entries: Debit "Customer Deposits" $100, Credit "Tokenized Deposits" $100
    - Mark Digital Lockbox funds as "pending mint" (reserved but not yet backed by on-chain deposit tokens)

At this point: The customer has $100 reserved in their Digital Lockbox (off-chain), but no on-chain deposit tokens exist yet.

Phase 2: On-Chain Minting (Blockchain Operations)

6. **Mint Instruction:** The Accounting Ledger instructs the Minter Service to mint tokens, providing: customer wallet address, amount ($100), and lockbox reference ID.

7. **On-Chain Mint Transaction:** The Minter Service submits a blockchain transaction calling `mint(customerAddress, 100)` on the Tokenized Deposit smart contract.

8. **Smart Contract Verification:** The contract's `_beforeTokenTransfer` hook verifies:

    - Identity NFT check: Does customer wallet hold an Identity NFT?
    - If yes: proceed with mint
    - If no: revert transaction (this should never happen if systems are working correctly)

9. **Token Balance Update:** The smart contract increases the customer's on-chain deposit token balance by 100 tokens and increases total supply by 100.

10. **Event Emission:** The contract emits a `Transfer` event with:

    - `from`: 0x0000...0000 (mint)
    - `to`: customer wallet address
    - `value`: 100

At this point: The blockchain shows 100 tokens in the customer's wallet. The Accounting Ledger still shows "pending mint" status.

Phase 3: Real-Time Reconciliation (Synchronization)

11. **Event Detection:** Block-Listener detects the mint event on the blockchain.

12. **Finality Wait:** Block-Listener monitors the blockchain until the transaction reaches finality (`n` confirmations, depending on the blockchain and network conditions). 12 confirmations would typically takes 2-3 minutes on Ethereum.

13. **Event Publication:** Block-Listener publishes a finalized mint event to the Event Broker containing: transaction hash, block number, customer wallet, amount, timestamp.

14. **Reconciliation Processing:** The Reconciliation Service consumes the event and:

    - Verifies the event matches a pending mint in the Accounting Ledger
    - Updates Digital Lockbox status from "pending mint" to "active" (now backing on-chain deposit tokens)
    - Logs the reconciliation with blockchain transaction hash as proof
    - Updates system totals: Total Active Digital Lockbox Balance += $100

15. **Continuous Verification:** Discrepancy Detection Service runs its check:

    - Query blockchain: `totalSupply()` on smart contract
    - Query Accounting Ledger: SUM(all active Digital Lockboxes)
    - Verify: these values are equal (zero discrepancy)
    - Log verification result with timestamp

16. **Completion:** Customer is notified that tokens are now available in their wallet for use in transfer of deposits or other transfers.

Critical Insight: The blockchain determines token existence and location. The Accounting Ledger determines reserve allocation and ownership mapping to legal identities. Reconciliation ensures these remain synchronized in real-time.

What If Scenarios

- **Blockchain transaction fails:** Minter Service detects failure, notifies Accounting Ledger, funds released from "pending" back to general account, customer notified
- **Reconciliation delay:** Tokens exist on-chain but show "pending confirmation" in customer interface until reconciliation completes
- **Discrepancy detected:** Emergency alerts triggered, minting halted, forensic investigation initiated, systems remain operational for transfers and burns

4.2 Withdrawal and Burning Flow

The withdrawal process enforces a strict "burn-then-release" mechanism to prevent double-spend conditions. This is the most security-critical flow because it converts on-chain deposit tokens back to fiat (see Figure 2):

Phase 1: Withdrawal Request (Accounting Ledger Operations)

1. **Withdrawal Initiation:** Customer initiates a withdrawal request through the customer interface, specifying: amount ($100), destination bank account, withdrawal reference ID.
2. **Verification:** The Accounting Ledger verifies:
   - Customer has active Digital Lockbox with sufficient balance ($100 available)
   - Customer's wallet address holds corresponding on-chain deposit tokens
   - No pending operations that would conflict with withdrawal
3. **Pre-Authorization:** If using `burnFrom()`, customer must have previously approved the Burner Service to burn tokens on their behalf (standard ERC-20 approval pattern). Most implementations use direct user-initiated burn to avoid this step.

Phase 2: on-chain deposit token Burn (Blockchain Operations)

4. **Burn Instruction:** The Accounting Ledger instructs the Burner Service to burn tokens, providing: customer wallet address, amount (100), withdrawal reference ID.
5. **On-Chain Burn Transaction:** The Burner Service submits a blockchain transaction calling `burnFrom(customerAddress, 100)` on the Tokenized Deposit smart contract.
6. **Token Destruction:** The smart contract:
   - Verifies the Burner Service has authorization (via approval or inherent contract logic)
   - Decreases the customer's on-chain deposit token balance by 100 tokens
   - Decreases total supply by 100 tokens
   - This operation is *irreversible*—once burned, tokens cannot be recovered
7. **Event Emission:** The contract emits a `Transfer` event with:
   - `from`: customer wallet address
   - `to`: 0x0000...0000 (burn)
   - `value`: 100

At this point: The tokens have been destroyed on-chain (permanently and irreversibly). The customer's Digital Lockbox still shows $100 reserved.

Phase 3: Real-Time Reconciliation (Synchronization)

8. **Event Detection:** Block-Listener detects the burn event on the blockchain.
9. **Finality Wait:** Block-Listener monitors the blockchain until the burn transaction reaches finality (12 confirmations).
10. **Event Publication:** Block-Listener publishes a finalized burn event to the Event Broker containing: transaction hash, customer wallet, amount, withdrawal reference ID.
11. **Reconciliation Processing:** The Reconciliation Service consumes the burn event and:
    - Verifies the event matches a pending withdrawal in the Accounting Ledger
    - Updates Digital Lockbox: reduces allocation by $100
    - Transfers $100 from Digital Lockbox to customer's general account
    - General Ledger entries: Debit "Tokenized Deposits" $100, Credit "Customer Deposits" $100
    - Marks withdrawal as "ready for payout"
    - Updates system totals: Total Active Digital Lockbox Balance -= $100
12. **Continuous Verification:** Discrepancy Detection Service runs its check:
    - Query blockchain: `totalSupply()` (should be 100 less than before)

- Query Accounting Ledger: SUM(all active Digital Lockboxes) (should be $100 less)
- Verify: these values remain equal (zero discrepancy)

Phase 4: Fiat Payout (External Banking Operations)

13. **Payout Initiation:** The Accounting Ledger instructs the USD Custodian to initiate wire transfer or ACH credit to customer's external bank account.
14. **Funds in Transit:**
    - Internal transfer: Customer General Account → Cash-in-Transit account
    - General Ledger entries: Debit "Customer Deposits" $100, Credit "Cash-in-Transit" $100
    - Wire transfer initiated to customer's external bank
15. **Settlement Confirmation:** When wire transfer confirms:
    - Cash-in-Transit account reduced by $100
    - General Ledger entries: Debit "Cash-in-Transit" $100, Credit "FBO Bank Account" $100
    - Customer notified of successful withdrawal

Critical Security Property: Burn-Then-Release The sequencing is deliberate and security-critical:

1. **First:** Tokens are burned on-chain (irreversible, public, immutable)
2. **Then:** Blockchain finality is achieved
3. **Only then:** Funds are released from Digital Lockbox

This ordering eliminates double-spend risk. A customer cannot hold both on-chain deposit tokens and redeemed fiat simultaneously. Even if there's a system failure after the burn but before reconciliation completes, the blockchain provides cryptographic proof that tokens were destroyed, enabling manual reconciliation and fund release.

What If Scenarios

- **Blockchain burn transaction fails:** Burner Service detects failure, notifies Accounting Ledger, withdrawal request marked as failed, customer retains tokens and can retry
- **Burn succeeds but wire transfer fails:** Customer's funds are in general account (no longer in Digital Lockbox), wire transfer retried, tokens already burned so customer cannot double-spend
- **Customer attempts to transfer tokens during withdrawal:** If burn hasn't executed yet, transfer succeeds normally; once burn executes, customer no longer has tokens to transfer
- **Reconciliation delay between burn and ledger update:** Customer may see "withdrawal processing" status; blockchain shows tokens are gone, but Accounting Ledger shows funds still in Digital Lockbox until reconciliation completes (conservative from customer perspective)

4.3 Peer-to-Peer Transfer with Reconciliation

Peer-to-peer transfers are the most common operation and demonstrate the elegant separation between token movement (blockchain) and reserve reallocation (Accounting Ledger). This flow also showcases the "public-chain-first" settlement model:

Phase 1: On-Chain Transfer (Blockchain Operations)

1. **Transfer Initiation:** Customer A uses any standard Ethereum wallet (MetaMask, Ledger, etc.) to initiate a transfer of 20 tokens to Customer B's wallet address. This is a standard ERC-20 `transfer()` call requiring only: recipient address, amount.
2. **Smart Contract Verification:** The Tokenized Deposit contract's `_beforeTokenTransfer` hook executes:
    - Query Identity NFT contract: `balanceOf(Customer A)` ¿ 0? (sender whitelisted?)
    - Query Identity NFT contract: `balanceOf(Customer B)` ¿ 0? (receiver whitelisted?)
    - If both checks pass: proceed with transfer

- If either check fails: revert transaction with error "Not whitelisted"

3. **Token Balance Update:** The contract:
   - Decreases Customer A's token balance: 100 → 80 tokens
   - Increases Customer B's token balance: 50 → 70 tokens
   - Total supply remains unchanged: 1,000 tokens
4. **Event Emission:** The contract emits a `Transfer` event:
   - `from`: Customer A wallet address
   - `to`: Customer B wallet address
   - `value`: 20

At this point: The blockchain shows Customer A has 80 tokens, Customer B has 70 tokens. The Accounting Ledger has not been updated yet—it still shows the old Digital Lockbox allocations.

Phase 2: Real-Time Reconciliation (Synchronization)

5. **Event Detection:** Block-Listener detects the transfer event on the blockchain immediately (within one block time).
6. **Finality Wait:** Block-Listener waits for finality. This is critical—we never process potentially reversible transactions.
7. **Event Publication:** Block-Listener publishes finalized transfer event to Event Broker with complete context: sender, receiver, amount, transaction hash, block number, timestamp.
8. **Reconciliation Processing:** The Reconciliation Service:
   - Identifies this as a peer-to-peer transfer (not mint/burn)
   - Looks up Customer A's Digital Lockbox in Accounting Ledger
   - Looks up Customer B's Digital Lockbox in Accounting Ledger
   - Executes atomic Accounting Ledger operation:
     - Customer A Digital Lockbox: $100 → $80 (-$20)
     - Customer B Digital Lockbox: $50 → $70 (+$20)
   - Note: General Ledger entries are NOT required—this is an internal reallocation within the "Tokenized Deposits" category
   - Logs the reconciliation with blockchain transaction hash as cryptographic proof
9. **Continuous Verification:** Discrepancy Detection Service verifies:
   - Blockchain total supply: unchanged at 1,000 tokens
   - Accounting Ledger total Digital Lockboxes: $100 - $20 + $50 + $20 = unchanged at $1,000
   - Invariant maintained: both still equal
10. **Notification:** Both customers receive notifications that the transfer has been finalized and reconciled.

Critical Architectural Principle: Blockchain as Transaction Ordering Authority This flow demonstrates the "public-chain-first" model:

- **The blockchain determines transaction validity and ordering:** Only after a transfer is final on-chain do we update the Accounting Ledger
- **Eliminates race conditions:** No possibility of the Accounting Ledger and blockchain showing conflicting states
- **Leverages economic security:** We inherit the security and battle-tested guarantees of the public blockchain relying solely on internal access controls
- **Provides cryptographic proof:** Every Accounting Ledger change is linked to an immutable blockchain transaction hash

What If Scenarios

- **Customer B is not whitelisted:** Transfer reverts on-chain immediately, Customer A retains tokens, no reconciliation needed
- **Transfer succeeds on-chain but reconciliation delayed:** Customers see different views temporarily:
  - Blockchain view (wallet): Customer A has 80 tokens, Customer B has 70 tokens (accurate)
  - Accounting Ledger view (bank portal): Customer A has $100, Customer B has $50 (stale, updates within 2-3 minutes)

  When in doubt, blockchain view is always authoritative

- **Rapid sequence of transfers:** Block-Listener processes events in blockchain order (block number, transaction index, log index), ensuring Accounting Ledger updates maintain correct sequencing
- **Chain reorganization before finality:** Block-Listener only processes finalized events, so reorganizations never affect the Accounting Ledger

Efficiency Note For peer-to-peer transfers, the Accounting Ledger update is purely informational—it maintains a consistent view for customer service and regulatory reporting. The blockchain is the sole source of truth for token ownership. Even if the Accounting Ledger experiences downtime, on-chain transfers continue functioning normally, with reconciliation catching up when the system recovers.

5 System Guarantees, Risk Mitigation, and Security

The architecture provides multiple layers of guarantees that work together to ensure system integrity. Understanding these guarantees requires clear recognition of the separation of concerns between blockchain and Accounting Ledger.

5.1 The Settlement Guarantee: Blockchain as Ultimate Arbiter

The "public-chain-first" settlement model is a deliberate design choice with profound security implications:

How It Works

- Every token state change (mint, burn, transfer) occurs first on the blockchain
- The Block-Listener waits for blockchain finality before publishing events
- Only after finality does the Reconciliation Service update the Accounting Ledger
- The Accounting Ledger never initiates balance changes—it only reacts to finalized blockchain events

Security Properties

- **Eliminates Race Conditions:** The Accounting Ledger state can never conflict with blockchain state because blockchain events are the sole source of truth for token movements
- **Leverages Economic Security:** Rather than relying solely on internal access controls, we inherit the economic security of the public blockchain (e.g., Ethereum's $50B+ staked value)
- **Provides Cryptographic Proof:** Every Accounting Ledger update is linked to an immutable blockchain transaction hash, providing undeniable proof of the state change
- **Prevents Internal Manipulation:** Even with full database access, an insider cannot create fraudulent Accounting Ledger entries without corresponding blockchain transactions

Contrast with Alternative Architectures Systems that settle on internal ledgers first, then later synchronize with blockchain state, introduce dangerous windows where internal and blockchain state can diverge. This creates:

- Reconciliation challenges when blockchain transactions fail
- Potential for double-spending if internal ledger grants provisional access to funds
- Complex rollback logic when blockchain and internal ledger conflict

Our architecture eliminates these issues entirely. The blockchain is not a secondary record—it is the primary source of truth for token state.

5.2 The Reserve Guarantee: Continuous Real-Time Verification

The reserve guarantee addresses the critical question: "Is every on-chain deposit token backed 1:1 by a real dollar in the bank?" We answer it with continuous, real-time verification:

The Fundamental Invariant The system maintains a mathematical invariant at all times:

$$\text{Blockchain totalSupply}() = \sum_{\text{all customers}} \text{AccountingLedger.DigitalLockbox}_i \tag{3}$$

This equation defines full reserve backing. The left side (blockchain) represents the liability (tokens in circulation). The right side (Accounting Ledger) represents the asset (segregated dollar reserves).

How Verification Works The Discrepancy Detection Service continuously verifies this invariant:

1. Every 60 seconds (configurable): Query blockchain for `totalSupply()`
2. Query Accounting Ledger database: `SELECT SUM(balance) FROM digital_lockboxes WHERE status='active'`
3. Compare the two values: `abs(blockchain_supply - ledger_total) == 0`
4. Log verification result with timestamp
5. If discrepancy detected (should never happen): halt minting, trigger emergency alerts

Transparency Properties

- **Publicly Verifiable Liability:** Anyone can query the blockchain to determine how many tokens exist. This is public, permissionless, and cryptographically guaranteed.
- **Auditable Asset Allocation:** Regulators and auditors have direct database access to verify the Accounting Ledger shows matching reserve allocations.
- **Immutable Reconciliation Trail:** The Event Store provides a complete audit log linking every blockchain event to every Accounting Ledger update, with timestamps proving real-time reconciliation.
- **Immediate Discrepancy Detection:** Unlike quarterly attestations, any discrepancy is detected within 60 seconds maximum, enabling immediate investigation and resolution.

Reconciliation as Proof-of-Reserves Traditional proof-of-reserves requires:

1. Verifying token supply (liabilities)
2. Verifying bank reserves (assets)
3. Confirming they match at a specific point in time

Our real-time reconciliation architecture provides this continuously:

1. Token supply is always verifiable on blockchain
2. Digital Lockbox reserves are always verifiable in Accounting Ledger
3. Discrepancy Detection Service verifies they match every minute
4. Event Store provides immutable proof this verification occurs continuously

This transforms proof-of-reserves from a periodic attestation into a continuous, automated, cryptographically-verifiable property of the system.

5.3 The Compliance Guarantee: Identity NFT Framework

The non-transferable Identity NFT acts as a durable, on-chain "passport." This framework provides a powerful compliance guarantee: value can only ever be held by or transferred between wallets that have been directly vetted and issued an NFT by our institution. This creates a closed-loop ecosystem on public infrastructure, enabling us to enforce sanctions and comply with judicial orders by revoking (burning) a user's Identity NFT, thereby immediately freezing their ability to transact.

Unlike systems that rely on centralized transaction approval, this framework places compliance verification directly on-chain, enabling standard wallet software to interact with Tokenized Deposits without requiring custom authorization protocols.

5.4 The Operational Guarantee: Fund Recoverability

Unlike bearer assets such as Bitcoin, the Tokenized Deposit is not subject to permanent loss from events like a compromised or lost private key. Because the Core Ledger maintains the ultimate record of ownership tied to a verified legal identity, we can offer a secure asset recovery process.

In the event of a key loss, a customer undergoes rigorous identity verification with our institution. Upon successful verification, we execute a recovery protocol: the Identity NFT and associated Tokenized Deposits tied to the compromised wallet are burned, and an equivalent amount is re-issued to a new, secure wallet provided by the customer. This provides a critical safety net for institutions and individuals, eliminating a major operational risk of tokenized deposit management.

5.5 Fault Tolerance and System Resilience

The event-driven architecture implements multiple layers of fault tolerance:

- **Circuit Breakers:** Prevent cascading failures by isolating failing components when error rates exceed thresholds.
- **Automatic Retry Logic:** Transient failures (network timeouts, temporary service unavailability) are automatically retried with exponential backoff, ensuring that temporary issues don't result in permanent processing failures.
- **Dead Letter Queues:** Events that fail processing after multiple retry attempts are routed to specialized queues for manual review, ensuring that problematic transactions are never lost and can be resolved by operations teams.
- **System Replay Capability:** The Event Store enables complete system state reconstruction from any point in time, supporting disaster recovery and enabling debugging of historical issues.
- **Graceful Degradation:** During partial system failures, critical operations (transfers, burn requests) continue functioning while non-essential operations (analytics, reporting) may operate with reduced functionality.

5.6 Addressing Core Banking Risks

The architecture of the Tokenized Deposit systematically eliminates the risks inherent to fractional-reserve digital banking:

- **Reserve Risk is Eliminated:** Full-reserve narrow banking means every Tokenized Deposit is backed 1:1 by cash or short-term U.S. Treasuries. Real-time reconciliation provides continuous cryptographic proof of this backing.
- **Counterparty Risk is Eliminated:** Users have a direct depository relationship with a regulated U.S. banking institution, not an unregulated offshore entity. This provides clear legal standing and eliminates ambiguity regarding the issuer's obligations.
- **Settlement Risk is Eliminated:** The full-reserve model is not subject to the Herstatt Risk that affects fractional-reserve institutions and does not depend on a government lender of last resort to ensure settlement.
- **Reconciliation Risk is Eliminated:** Event-driven architecture eliminates the batch processing windows that create temporary inconsistencies in traditional banking systems. Every transaction is reconciled in near real-time.

5.7 System Constraints and Considerations

We acknowledge several architectural constraints that represent trade-offs inherent to the design:

- **Transaction Costs:** Users pay network gas fees for on-chain transactions. While this ensures access to decentralized settlement, it introduces cost volatility during network congestion. Future roadmap items include gas-relayer services and Layer 2 deployment to reduce and stabilize costs.
- **Data Privacy:** Transaction details (sender, receiver, amount) are publicly visible on the blockchain. This transparency is intentional for verifying reserves but may not suit all use cases. Future upgrades may incorporate zero-knowledge proofs for confidential transfers.
- **Throughput Constraints:** System throughput is limited by the underlying blockchain's capacity. While sufficient for most treasury and deposit-transfer use cases, high-frequency trading applications may require Layer 2 solutions.
- **Centralized Governance:** Identity NFT issuance and Core Ledger administration are centralized functions managed by our institution. This is necessary for regulatory compliance and ensures clear accountability, but represents a trust assumption for users.

6 Real-Time Reconciliation vs. Batch Processing: A Paradigm Shift

Traditional bank reconciliation operates on a batch processing model: transactions are collected over time (hourly, daily, or longer) and processed together in scheduled batches. This approach, while computationally efficient for historical data, introduces critical weaknesses that are unacceptable for a blockchain-native tokenized deposit.

6.1 The Batch Processing Problem

Temporal Windows of Inconsistency Batch systems create windows—often hours or days—where different systems show conflicting states:

- Blockchain shows tokens exist and have been transferred
- Internal accounting systems show old balances
- Customer service representatives see stale data
- Regulatory reporting lags behind actual positions

During these windows, several risks emerge:

Delayed Fraud Detection Fraudulent activity may not be detected until the next batch run, potentially allowing:

- Unauthorized minting to go unnoticed for hours
- Discrepancies between on-chain supply and reserves to persist
- Multiple fraudulent transactions before detection and system halt

Regulatory Compliance Challenges Regulators increasingly demand real-time visibility into financial positions. Batch processing cannot provide:

- Instantaneous reserve ratio reporting
- Real-time detection of suspicious transaction patterns
- Immediate compliance with sanctions list updates
- Accurate point-in-time balance snapshots for regulatory stress tests

Customer Experience Degradation Modern customers expect immediate confirmation. Batch processing creates:

- Delayed balance updates in customer interfaces
- Uncertainty about whether transactions have truly settled
- Support burden from customers seeing inconsistent balances across systems

6.2 The Real-Time Reconciliation Advantage

Our event-driven reconciliation architecture eliminates these problems:

Continuous Consistency

- Accounting Ledger updates occur within minutes of blockchain finality
- Discrepancy detection runs every 60 seconds
- No temporal windows where systems show fundamentally different states
- Customer-facing systems always reflect near-current blockchain state

Immediate Threat Detection

- Unauthorized minting detected within 60 seconds maximum
- Automatic system halt prevents cascading fraudulent transactions
- Forensic investigation begins immediately with full event trail
- Minimal window for damage from security breaches

Enhanced Regulatory Compliance

- Real-time reserve ratio verification
- Continuous audit trail with blockchain-backed proof
- Immediate sanctions enforcement through Identity NFT burning
- Point-in-time state reconstruction for any past moment
- Regulatory reporting generated on-demand with current data

Superior Operational Resilience

- Event-driven architecture enables graceful degradation
- Failed reconciliation attempts automatically retry
- Event Store enables complete state reconstruction after failures
- System can process events in parallel for high throughput

6.3 Architectural Enablers

Real-time reconciliation is made possible by specific architectural decisions:

Blockchain as Event Source Using the blockchain as the authoritative event source provides:

- Immutable, ordered event stream
- Cryptographic proof of every state change
- Deterministic event ordering (block number, transaction index, log index)
- No possibility of event loss or tampering

Event-Driven Processing Modern event streaming platforms (Kafka, Kinesis) enable:

- Sub-second event delivery latency
- Guaranteed event ordering and delivery
- Parallel processing across multiple customers
- Automatic retry and fault tolerance

Eventual Consistency Model Rather than requiring immediate ACID consistency across all systems:

- Blockchain is immediately consistent (transaction finality)
- Accounting Ledger converges to consistency within minutes
- System maintains monotonic progress (no rollbacks)
- Temporary inconsistencies are bounded and observable

6.4 Performance Characteristics

Metric	Batch Processing	Real-Time Reconciliation
Reconciliation Latency	Hours to Days	2-5 minutes
Discrepancy Detection	Next batch cycle	¡ 60 seconds
Fraud Detection Window	Hours	Minutes
Customer Balance Updates	Delayed	Near real-time
Regulatory Reporting	Stale data	Current data
System Recovery	Complex	Event replay
Audit Trail Quality	Batch logs	Immutable event stream

7 Comparison to Alternative Approaches

7.1 Versus Blockchain-Based Bank Deposits

Some institutions are exploring tokenized bank deposits on permissioned blockchains (e.g., JPMorgan's JPM Coin, Signature's Signet). These systems tokenize fractional-reserve deposits, inheriting traditional banking risks. The Tokenized Deposit provides:

- **Full-Reserve Backing:** Eliminates fractional-reserve settlement risk (Herstatt Risk). No reliance on government lender of last resort for settlement assurance.
- **Public Blockchain Access:** Interoperability with cross-border transfer of deposits and the broader ecosystem versus siloed private networks.
- **Transparent Verification:** Public blockchain enables anyone to verify token supply; real-time reconciliation provides continuous reserve proof versus opaque internal systems.
- **Real-Time vs. Batch Reconciliation:** While permissioned deposit tokens may use batch processing for internal reconciliation, our event-driven architecture provides minute-by-minute verification.
- **Economic Security:** Leverages the $50B+ staked value securing Ethereum versus relying solely on internal database access controls.

7.2 Versus Central Bank Digital Currencies (CBDCs)

CBDCs represent government-issued digital currencies with different design goals (monetary policy, financial inclusion). The Tokenized Deposit serves a complementary role:

- **Private Sector Innovation:** Market-driven features and integrations
- **Immediate Availability:** Operational now versus multi-year CBDC timelines
- **Blockchain-Native Design:** Built for programmability

8 Vision for the Future

8.1 The Three Pillars of Ideal Infrastructure for Transfer of Deposits

An ideal system for transfer of deposits has always aspired to deliver three goods: **broad acceptance**, **no credit risk**, and **finality of settlement**. Legacy systems, with their reliance on fractional reserves and built-in settlement lags, fail on the latter two points.

The Tokenized Deposit, operating within a full-reserve narrow bank environment with real-time reconciliation, uniquely provides all three pillars. By doing so, it will unlock new efficiencies for both digital and real economies.

8.2 Enabling the Next Generation of Financial Infrastructure

The combination of blockchain settlement and real-time reconciliation enables capabilities impossible with traditional infrastructure:

- **Atomic Multi-Party Transactions:** Smart contracts can coordinate complex transfers of deposits across multiple parties with instant settlement and cryptographic proof of completion. Real-time reconciliation ensures all parties' reserve allocations update within minutes, providing immediate confirmation of settlement finality.
- **Programmable Treasury Operations:** Enterprises can implement sophisticated cash management strategies using smart contracts, with real-time visibility into all positions. The Accounting Ledger provides traditional accounting views while blockchain provides programmable execution—reconciled continuously.
- **Instant Securities Settlement:** Delivery-versus-settlement for securities trades can occur atomically on-chain, eliminating the T+X settlement lag and associated credit risk. Real-time reconciliation provides immediate regulatory reporting of positions, enabling same-day settlement with full audit trail.
- **Cross-Border Transfer of Deposits Without Intermediaries:** Deposits can be transferred directly between parties on public infrastructure, eliminating correspondent banking fees and delays. Real-time reconciliation provides both source and destination banks with immediate visibility into reserve movements, satisfying regulatory requirements for cross-border transaction monitoring.
- **Transparent Supply Chain Finance:** Settlement terms can be encoded in smart contracts with automatic execution upon delivery confirmation. Real-time reconciliation ensures all parties (buyer, seller, financiers) see consistent account states within minutes, reducing disputes and improving working capital efficiency.
- **Regulatory Reporting in Real-Time:** Rather than filing periodic reports based on stale data, institutions can provide regulators with live dashboards showing current positions, reserve ratios, and transaction flows—all backed by continuous reconciliation proof.
- **Instant Audit Trails:** Every financial event is linked to an immutable blockchain transaction hash and logged in the Event Store. Auditors can verify any past state, trace any transaction through complete lifecycle, and confirm continuous reconciliation occurred—all without manual sampling or trust in reconstructed records.

8.3 Building Capital Markets for the Internet Age

The Tokenized Deposit is more than a new bank rail; it is foundational infrastructure for rebuilding capital markets and commerce from the ground up, at internet scale. By providing the stability of a bank deposit, the programmability of a blockchain-native representation of deposits, and continuous cryptographic proof of reserves through real-time reconciliation, it eliminates the fundamental trade-offs that have constrained transfer of deposits.

The architectural innovation is not merely using blockchain for transfer of deposits—many systems do that. The innovation is the real-time, event-driven reconciliation architecture that maintains continuous synchronization between blockchain token state and bank reserve allocation, providing:

- **Cryptographic Proof:** Every reserve allocation linked to blockchain transaction hash
- **Continuous Verification:** 1:1 backing verified every block, not quarterly
- **Immediate Detection:** Discrepancies caught in seconds, not days
- **Operational Resilience:** Event-driven architecture enables fault tolerance and replay
- **Regulatory Compliance:** Real-time reporting with immutable audit trail

Complex transactions occur transparently without the settlement risk inherent to fractional banking. The result is a financial system that operates at the speed and scale of the internet while maintaining the security, transparency, and regulatory certainty that institutional participants require.

This is not a future vision—this architecture is operational and proven. The Tokenized Deposit demonstrates that we can have both the programmability of blockchain and the security of traditional banking, without compromise, through the disciplined application of real-time reconciliation principles.

References

[1] Financial Accounting Standards Board (FASB). (2023). *Accounting for and Disclosure of Crypto Assets (ASU 2023-08)*. Retrieved from https://www.fasb.org/projects/completed/accounting-for-and-disclosure-of-crypto-assets

[2] Bank for International Settlements (BIS). (1974). *Annual Report: 1974*. Retrieved from https://www.bis.org/publ/arpdf/archive/ar1974.pdf

[3] Buterin, V., & Vogelsteller, F. (2015). *EIP-20: Token Standard*. Ethereum Improvement Proposals. Retrieved from https://eips.ethereum.org/EIPS/eip-20

[4] Entriken, W., Shirley, D., Evans, J., & Sachs, N. (2018). *EIP-721: Non-Fungible Token Standard*. Ethereum Improvement Proposals. Retrieved from https://eips.ethereum.org/EIPS/eip-721

[5] Brewer, E. (2012). *CAP Twelve Years Later: How the 'Rules' Have Changed*. IEEE Computer, 45(2), 23-29.

[6] Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

[7] Fowler, M. (2017). *Event-Driven Architecture*. Retrieved from https://martinfowler.com/articles/201701-event-driven.html

[8] TrueDigital. (2019). *TrueDigital Holdings Launches Signet*. Retrieved from https://web.archive.org/web/20190121170648/https://www.truedgtl.com/truedigital-launches-signet

[9] The Currency Analytics. (2021). *Replacing Wire Transfers With Blockchain: Signet and JP Morgan*. Retrieved from https://thecurrencyanalytics.com/blockchain/replacing-wire-transfers-with-blockchain-signet-and-jp-morgan-4659

[10] Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Retrieved from https://bitcoin.org/bitcoin.pdf