LIQUIDITY.IO

Liquid Securities

Compliant Digital Securities on EVM
with Transfer Restrictions and Corporate Actions

Zach Kelling

Satschel, Inc.

Version 1.0 — April 2026

ERC-3643 (T-REX) · ERC-1404 Transfer Restrictions
Reg D 506(b)/506(c) · Rule 144 Lockups · On-Chain Cap Table
Dividends · Stock Splits · Mergers · KYC/AML Gating

Contents

Abstract

Liquid Securities is a token standard for issuing and managing compliant digital securities on the Liquidity Network (chain ID 8675309). Built on ERC-3643 (T-REX) and ERC-1404, the standard provides on-chain transfer restrictions, KYC/AML whitelisting, corporate action execution, and cap table management. This paper specifies the smart contract architecture, compliance modules, and corporate action lifecycle for digital securities issued on the Liquidity.io ATS.

Introduction

Digital securities—tokenized representations of traditional financial instruments such as equity, debt, and fund interests—require a fundamentally different token standard than utility tokens. A compliant digital security must enforce:

- **Transfer restrictions**: Not every holder can transfer to every recipient at any time.
- **Identity verification**: Both sender and receiver must be KYC-verified and accredited (if required).
- **Jurisdictional compliance**: Transfers must respect jurisdiction-specific regulations and sanctions lists.
- **Holding period enforcement**: Restricted securities under Rule 144 cannot be resold before the holding period expires.
- **Corporate actions**: Dividends, stock splits, reverse splits, mergers, and forced transfers (e.g., court orders) must be executable by the issuer or transfer agent.

The Liquid Securities standard addresses all of these requirements with a modular, composable architecture deployed natively on the Liquidity Network.

Relationship to Existing Standards

Standard	Origin	Role in Liquid Securities
ERC-3643 (T-REX)	Tokeny [1]	Core token + identity registry + compliance
ERC-1404	TokenSoft	Transfer restriction codes (human-readable errors)
ERC-20	Ethereum	Base token interface (balanceOf, transfer, approve)
ERC-2612	Ethereum	Gasless approvals (permit)

Token Architecture

Contract Hierarchy

```
LiquidSecurity (ERC-3643 Token)
  |-- IdentityRegistry KYC/AML identity storage
  | |-- ClaimTopics Required claim types per token
  | |-- TrustedIssuers Approved KYC providers
  |-- ComplianceModule Transfer restriction logic
  | |-- JurisdictionModule Country-level allow/block
  | |-- AccreditationModule Investor qualification
  | |-- LockupModule Rule 144 holding periods
```

```
  | |-- MaxHoldersModule Reg D investor count limits
  |-- CorporateActions Dividends, splits, mergers
  |-- CapTable Shareholder registry
  |-- TransferAgent SEC-registered TA functions
```

Token Lifecycle

1. **Issuance**: Issuer deploys `LiquidSecurity` with compliance modules configured. Transfer agent mints tokens to verified investors.
2. **Trading**: Transfers route through the compliance module. Each transfer is checked against all active restriction modules. If any module rejects, the transfer reverts with an ERC-1404 restriction code.
3. **Corporate Actions**: The transfer agent or issuer executes dividends, splits, or forced transfers through the `CorporateActions` contract.
4. **Redemption**: Issuer can burn tokens (e.g., bond maturity, share buyback) through the transfer agent role.

Compliance Modules

Module Interface

Every compliance module implements a single interface:

```
interface IComplianceModule {
    function checkTransfer(
        address from,
        address to,
        uint256 amount,
        bytes32 tokenId
    ) external view returns (uint8 restrictionCode);

    function messageForTransferRestriction(uint8 code)
        external view returns (string memory);
}
```

Restriction code 0 means "transfer allowed." Any non-zero code causes the transfer to revert, and the message function returns a human-readable explanation (ERC-1404).

Jurisdiction Module

Controls which countries can hold and transfer the security:

```
JurisdictionModule {
    mapping(bytes2 => bool) allowedCountries; // ISO 3166-1 alpha-2
    mapping(bytes2 => bool) blockedCountries; // OFAC sanctions
    mapping(bytes2 => uint256) maxHoldersByCountry;
}
```

Restriction codes:

Code	Message
1	Sender country not allowed

2	Receiver country not allowed
3	Country holder limit reached
4	Country on sanctions list

Accreditation Module

Enforces investor qualification requirements per Regulation D:

Regulation	Requirement	On-Chain Check
Reg D 506(b)	≤35 non-accredited investors	Counter per offering
Reg D 506(c)	All investors accredited	`isAccredited(address)`
Reg S	Non-US persons only	`country != "US"`
Reg A+	Tier 2: $75M max	Cumulative amount tracker

Accreditation status is stored in the Identity Registry as a verifiable claim issued by a trusted KYC provider (e.g., Parallel Markets, Verify Investor). Claims have expiry dates; expired accreditation blocks transfers until renewed.

Lockup Module (Rule 144)

SEC Rule 144 [3] requires a holding period before restricted securities can be resold:

```
LockupModule {
    struct Lockup {
        uint256 startTime; // Acquisition timestamp
        uint256 duration; // 6 months (reporting) or 12 months
        bool isAffiliate; // Affiliate restrictions apply
        uint256 volumeLimit; // 1% of outstanding or avg weekly volume
    }

    mapping(address => mapping(bytes32 => Lockup)) lockups;
}
```

Rule 144 conditions checked on-chain:

1. Holding period satisfied (6 months for reporting issuers, 12 months otherwise)
2. Volume limitation (affiliates: ≤1% of outstanding shares per quarter)
3. Manner of sale (executed through the ATS or broker-dealer)
4. Current public information available (issuer is SEC-reporting)
5. Form 144 filing (triggered off-chain, recorded on-chain)

Max Holders Module

Enforces maximum shareholder counts required by certain exemptions:

Rule	Limit
Reg D 506(b)	35 non-accredited investors
Section 12(g)	2,000 holders (or 500 non-accredited) before SEC reporting

Identity Registry

Architecture

The Identity Registry is a permissioned on-chain store of verified investor identities. It follows the ERC-3643 ONCHAINID standard:

```
IdentityRegistry {
    mapping(address => Identity) identities;

    struct Identity {
        bytes2 country; // ISO 3166-1 alpha-2
        bool isAccredited; // Accredited investor status
        uint256 accreditedUntil; // Expiry timestamp
        uint8 investorType; // 1=individual, 2=entity, 3=trust
        bytes32[] claims; // Verifiable claim hashes
    }

    address[] trustedIssuers; // Approved KYC/AML providers
    uint256[] requiredClaimTopics; // e.g., KYC=1, AML=2, ACCRED=3
}
```

Claim Topics

Topic ID	Name	Provider
1	KYC Verification	Parallel Markets, Persona
2	AML Screening	Chainalysis, Elliptic
3	Accredited Investor	Verify Investor, Parallel Markets
4	Qualified Purchaser	Manual attestation + legal opinion
5	Tax Residency	W-8/W-9 collection service

Privacy

Personally identifiable information (PII) is never stored on-chain. The Identity Registry stores only:

- Country code (2 bytes)
- Boolean flags (accredited, investor type)
- Claim hashes (SHA-256 of off-chain verifiable credentials)

Full identity data is stored off-chain in the Liquidity.io KMS-encrypted identity vault, accessible only by the transfer agent and compliance officer with appropriate credentials.

Corporate Actions

Supported Actions

Action	Trigger	On-Chain Effect	Authorization
Cash Dividend	Board resolution	USDL transfer pro-rata	Transfer Agent
Stock Dividend	Board resolution	Mint tokens pro-rata	Transfer Agent

Forward Split	Board resolution	Multiply balances by ratio	Transfer Agent
Reverse Split	Board resolution	Divide balances, round	Transfer Agent
Merger	Board + shareholder vote	Swap tokens for acquirer	Transfer Agent
Forced Transfer	Court order	Transfer between addresses	Transfer Agent
Share Buyback	Board resolution	Transfer to treasury, burn	Issuer + TA
Rights Offering	Board resolution	Mint new tokens to holders	Transfer Agent

Dividend Distribution

```
function distributeDividend(
    address token,
    address paymentToken, // USDL
    uint256 totalAmount,
    uint256 recordDate // Snapshot block
) external onlyTransferAgent {
    uint256 supply = IERC20(token).totalSupplyAt(recordDate);
    address[] memory holders = capTable.holdersAt(recordDate);

    for (uint i = 0; i < holders.length; i++) {
        uint256 balance = IERC20(token).balanceOfAt(
            holders[i], recordDate
        );
        uint256 payment = (totalAmount * balance) / supply;
        IERC20(paymentToken).transfer(holders[i], payment);
    }
}
```

The `recordDate` parameter uses ERC-20 snapshot functionality to determine balances at a specific block, preventing ex-dividend gaming.

Stock Split

Forward and reverse splits modify all balances atomically in a single transaction:

```
function executeSplit(
    address token,
    uint256 numerator, // e.g., 2 for 2:1 split
    uint256 denominator // e.g., 1 for 2:1 split
) external onlyTransferAgent {
    // Adjusts internal multiplier; no individual transfers needed
    LiquidSecurity(token).setSplitRatio(numerator, denominator);
    // All balanceOf() calls now return adjusted amounts
}
```

The split is implemented as a global multiplier rather than individual balance modifications, making it $O(1)$ regardless of holder count.

Cap Table Integration

On-Chain Cap Table

The cap table contract maintains a complete shareholder registry synchronized with token balances:

```
CapTable {
    struct Holder {
        address wallet;
        bytes32 identityHash; // Link to Identity Registry
        uint256 shareClass; // Common, Preferred A, B, etc.
        uint256 vestingStart;
        uint256 vestingCliff;
        uint256 vestingDuration;
        uint256 vestedAmount;
    }

    mapping(address => Holder) public holders;
    address[] public holderList;

    // Snapshot support for record dates
    mapping(uint256 => mapping(address => uint256)) balanceSnapshots;
}
```

Share Classes

Each security token can represent multiple share classes with different rights:

Class	Voting	Dividend Priority	Liquidation
Common	1 vote/share	Last	Last
Preferred A	None	1x before common	1x preference
Preferred B	None	1x before A	1x + participation
Founder	10 votes/share	Last	Last

Vesting Schedules

Employee equity grants enforce vesting on-chain:

- **Cliff**: Tokens are non-transferable until cliff date (typically 1 year)
- **Linear vesting**: After cliff, tokens vest linearly over the remaining period
- **Acceleration**: Double-trigger acceleration on change of control
- **Clawback**: Unvested tokens can be reclaimed by the issuer on termination

Deployment and Integration

Issuance Workflow

1. Issuer engages Satschel, Inc. as transfer agent and ATS operator.
2. Legal counsel prepares offering documents (PPM for Reg D, offering circular for Reg A+).
3. Transfer agent deploys `LiquidSecurity` token with configured compliance modules.
4. Investors complete KYC/AML through the Liquidity.io onboarding portal.
5. Verified investors are registered in the on-chain Identity Registry.
6. Transfer agent mints tokens to verified investor addresses.
7. Secondary trading begins on the Liquidity.io ATS.

Integration with Liquidity.io ATS

The Liquid Securities standard is natively integrated with the Liquidity.io trading platform:

Component	Integration
Order Matching	CLOB precompile (LP-9020) checks compliance before fill
Settlement	Atomic swap: security token ↔ USDL in one transaction
Custody	MPC wallet (CGGMP21) holds security tokens
Reporting	Blockscout indexes all transfers for regulatory reporting
Cap Table	Real-time shareholder registry for issuer portal

Deployed Contracts (Liquidity Network)

Contract	Address
IdentityRegistry	`0x1000000000000000000000000000000000000001`
ComplianceModule (proxy)	`0x1000000000000000000000000000000000000002`
CorporateActions	`0x1000000000000000000000000000000000000003`
CapTable (proxy)	`0x1000000000000000000000000000000000000004`
TransferAgent	`0x1000000000000000000000000000000000000005`

These are protocol-level singletons deployed at deterministic addresses on the Liquidity Network (chain ID 8675309). Individual security tokens reference these shared contracts.

References

[1] Tokeny, "ERC-3643: T-REX Token for Regulated Exchanges," Ethereum, 2021.

[2] TokenSoft, "ERC-1404: Simple Restricted Token Standard," Ethereum, 2018.

[3] SEC, "Rule 144: Selling Restricted and Control Securities," 17 CFR 230.144, 1972 (amended 2008).

[4] SEC, "Regulation D: Rules Governing the Limited Offer and Sale of Securities," 17 CFR 230.500–508, 1982 (amended 2020).

[5] SEC, "Regulation A: Conditional Small Issues Exemption," 17 CFR 230.251–263, 2015 (amended 2020).

[6] SEC, "Regulation S: Rules Governing Offers and Sales Made Outside the United States," 17 CFR 230.901–905, 1990.

[7] SEC, "Section 12(g): Registration Requirement for Securities," Securities Exchange Act of 1934, § 12(g).

[8] Z. Kelling, "Lux Standard: Smart Contract Library for EVM Chains," Lux Partners, 2024.

[9] Z. Kelling, "Lux Cap Table: On-Chain Shareholder Registry," Lux Partners, 2025.

[10] Z. Kelling, "Liquidity.io Full Stack Architecture," Satschel, Inc., 2026.

[11] Z. Kelling, "Lux Consensus: A Scalable BFT Protocol for Multi-Chain Networks," Lux Partners, 2023.

[12] Z. Kelling, "The Liquidity Network: A Regulated Securities Settlement Blockchain," Satschel, Inc., 2026.

[13] Tokeny, "ONCHAINID: Self-Sovereign Identity for Tokenized Securities," 2021.

The Liquid Securities standard is built on open source contracts from the Lux Network ecosystem. All upstream Lux repositories are open source (BSL-1.1, MIT, Apache-2.0).

Lux Platform Specs (LPs): https://github.com/luxfi/lps