

Report:

Tokens, Ethereum and Non

Ethereum Token Standards,

Real World Use Cases

Prepared By: Abhilash Rajan
Date: 04 October 2025

Tokens

Tokens serve as **fungible stores of value** that can be transferred P2P without a bank or middlemen. A token is **needed to pay for the usage of the network**. An integral part of the incentive scheme. The stakeholders of the blockchain network (**miners/validators or token holders**) are incentivized to contribute to the network by **being rewarded with a stake in the network**. Tokens rely on smart contracts for logic automation, ensuring decentralized execution of trades, payments, governance, and compliance.

The tokens work just like a restaurant coupon represents a meal. Tokens can be used to **represent a specific amount of physical or digital assets** that you could own, transfer, or redeem later.

Tokenization is the process of creating and issuing tokens. **Types of tokens**.

Security Tokens: They **cannot be freely transferred, has to follow regulatory guidelines or trade laws**. Owning a security token of a company means that you have rights over that company. Eg: **Shares**.

Utility Tokens: Offered by a firm or organization to use their products. Only applicable inside the corresponding firm. Eg: **Basic Attention Tokens (BAT), Uniswap (UNI)**.

Asset Tokens: Represents real assets like houses, gold, or bonds that can be traded.

Currency Tokens: We can use currency tokens in the same way we are using fiat currencies. Just like how we use fiat currency for purchasing, we can use currency tokens with merchants who accept them. **Ether is an example** of a currency token.

Governance Tokens: Provide voting rights in decentralized platforms (Eg: **Uniswap UNI, Maker MKR**).

Fungible and Non Fungible

Fungibility is the **feature of goods or assets to be merged, split, and interchanged** with each other. An example is **fiat currency**. A higher denomination currency note may be replaced with an equivalent value of lower denomination notes. Both have the same value.

Non-fungible tokens are **non-replicable and non-divisible**. Therefore non-fungible tokens are used to **represent collectibles and unique assets like identity, certificates, etc.** For example, let's consider the case of a **birth certificate, which is a unique asset** with an owner. You cannot exchange your birth certificate with another person's birth certificate.

Pros -

1. Fractional ownership **increases liquidity and lowers investment barriers.**
2. Automation via smart contracts **reduces transaction and management costs.**
3. Transparency and immutability **increase trust and reduce fraud.**
4. **Enables programmable and composable business logic** within decentralized systems.

Cons -

1. Legal and regulatory uncertainty, especially **across different jurisdictions**.
2. Potential for **liquidity/maturity mismatch and redemption run risks**.
3. Smart contract **vulnerabilities and security issues can lead to losses**.
4. Complexity in valuation and risk management for certain tokenized assets.

Legal Aspects of Tokens-

1. Token classification (utility, security, asset-backed) determines applicable regulations, most notably **regarding the securities law (SEC, FCA, ESMA etc.)**.
2. **KYC/AML (Know Your Customer/Anti-Money Laundering) compliance is required** for asset-backed and financial tokens.
3. Legal enforceability of smart contracts is **still evolving**, requiring dual agreements (on-chain and legal).
4. Howey Test (in the U.S.) **distinguishes security tokens from utility tokens and applies corresponding legal obligations**.

Token Standards

Standardization allows developers to launch tokens easily and fosters liquidity, utility, and innovation and provides specific features & functionalities for various use cases, ensuring smooth integration & interoperability within digital ecosystems. ERC Protocol is an official way of introducing certain improvement proposals to the Ethereum network. **ERC stands for Ethereum Request for Comments.**

- **ERC-20:** Used for fungible utility tokens, governance tokens, stablecoins, and liquidity tokens on the Ethereum blockchain, ensuring that tokens created on Ethereum can interact seamlessly with wallets, exchanges and dApps. **ERC-20 tokens use smart contracts to enable transferability, tracking of balances, and integration across Ethereum-based platforms;** include USDT, Chainlink, and UNI.
- **ERC-721:** Works in Ethereum Blockchain. Specialized for unique tokens (NFTs) like art, collectibles, in-game assets; interacted with via NFT marketplaces and wallets that support NFTs.
- **ERC-1155:** Works in Ethereum. Used for games and platforms requiring both fungible and non-fungible items; enables batch operations, leading to reduced transaction costs. Efficient batch transfers, supports both types.
- **ERC-777:** Works in Ethereum. Offers advanced smart contract interactions and improved security for token operations; ideal for complex dApps. Deals with fungible tokens. Backward compatible with ERC-20, & prevents certain issues.
- **BEP-20, TRC-20, SPL, FA1.2/FA2:** Blockchain-specific standards for tokens on BNB Chain, TRON, Solana, and Tezos, respectively. Each facilitates asset issuance, transfer, and integration with dApps on those platforms, at low fees, high speed etc.

Real-World Use Cases and Details

- **RealT:** Tokenizes **real estate**, allowing fractional investments and increased market liquidity.
- **IBM Food Trust:** Uses tokens for **supply chain transparency** and **food product tracking**.
- **Aave/Uniswap:** DeFi platforms use **governance and liquidity tokens** to facilitate decentralized finance operations like **lending and trading**.
- **BanQu:** Tokens promote financial inclusion via **microloans and digital IDs**.
- **Propy:** Tokenizes **real estate transactions**, ensuring **secure, fraud-resistant property transfers**.
- **Loyal:** Loyalty tokens let **companies and users enjoy flexible, tradable reward systems**.
- **Aventus:** Uses tokens for **secure event ticketing**, preventing fraud and scalping.
- **LBRY:** Content creators & consumers trade tokens for **decentralized publishing & monetization**.
- **SDX:** To **tokenize bonds and securities**, enabling **institutional-grade digital asset markets**.
- **HQLAx:** To tokenize **high-quality liquid assets**, improving **collateral mobility and settlement**.
- **RedSwan:** Tokenizes **commercial institutional assets** accessible via digital tokens.
- **Ripple / sgBENJI:** To tokenize **money market funds**, enabling **real-time trading and liquidity**.
- **Ripple Stablecoin (RLUSD):** Enabling fast and **regulated digital payments**.
- **Agora DCM:** To **digitize bond issuance & syndication**, reducing reconciliation, transaction delays.
- **Circle (USDC):** For stable, **regulated value transfer** and **asset-backed finance**.
- **Tether (USDT):** For **global liquidity and remittances**.
- **Paxos:** Issues **regulated stablecoins** on Ethereum for **asset settlement, payments, and tokenization**.

Project: RealIT - RealToken Inc. (USA)

Blockchain Platform: Ethereum

Token Standards / Types: ERC-20 (fractional ownership), ERC-721 (property NFTs)

Purpose: Fractional real estate investment and increased liquidity

Developer: RealToken Co-founded by Remy and Jean-Marc Jacobson



<https://realt.co>

Project: IBM Food Trust - IBM Corporation

Blockchain Platform: Hyperledger Fabric (private, permissioned)

Token Standards / Types: Custom tokens / digital assets

Purpose: Supply chain transparency and product tracking (especially in food industry)

Developer: IBM Blockchain Division, developed in collaboration with Walmart, Nestlé, and others



<https://byteally.com/insights/supply-chain/integrating-with-ibm-food-trust-blockchain-guide/>

Project: Aave - Aave Companies (formerly ETHLend)

Blockchain Platform: Ethereum

Token Standards / Types: ERC-20 (AAVE governance token, aTokens)

Purpose: Decentralized lending and borrowing

Developer: Founded by Stani Kulechov; Aave Protocol is open-source



<https://aave.com/>



UNISWAP

<https://app.uniswap.org/>

Project: Uniswap - Uniswap Labs

Blockchain Platform: Ethereum

Token Standards / Types: ERC-20 (UNI governance token, LP tokens)

Purpose: Decentralized trading and liquidity provisioning

Developer: By Hayden Adams, backed by Ethereum Foundation and venture funds

Project: **BanQu** - BanQu, Inc.

Blockchain Platform: **Ethereum-compatible / Hyperledger**

Token Standards / Types: Custom digital tokens

Purpose: **Financial inclusion via identity management and microfinance**

Developer: Founded by Ashish Gadnis; partners include AB InBev and United Nations



Project: **Propy** - Propy Inc.

Blockchain Platform: **Ethereum**

Token Standards / Types: ERC-721 (real estate NFTs)

Purpose: **Secure, tokenized real estate transactions and title transfers**

Developer: Founded by Natalia Karayaneva; partnered with governments and real estate agencies



Project: **Loyyal** - Loyyal Corporation

Blockchain Platform: **Ethereum + Hyperledger Fabric**

Token Standards / Types: ERC-20 (loyalty tokens)

Purpose: **Interoperable, tradable loyalty and rewards programs**

Developer: Based in the U.S.; enterprise clients include Emirates and Deloitte



Project: **Aventus** - Aventus Network Services Ltd.

Blockchain Platform: **Ethereum + Aventus Network (Layer 2)**

Token Standards / Types: ERC-20 (AVT token)

Purpose: **Secure event ticketing, anti-scalping, fraud prevention**

Developer: Founded by Alan Vey and Annika Monari; integrates with Ticketmaster and others



Project: **LBRY** - LBRY Inc.

Blockchain Platform: **LBRY custom blockchain**

Token Standards / Types: LBC (native token, non-ERC)

Purpose: **Decentralized publishing and monetization for creators**

Developer: Founded by Jeremy Kauffman; also operates Odysee (a LBRY-based platform)



<https://lbry.com/>

Project: **SDX** - SIX Digital Exchange (Switzerland)

Blockchain Platform: **Corda (by R3)**

Token Standards / Types: Custom tokens / smart contracts

Purpose: **Tokenized bonds and digital securities**

Developer: Developed in partnership with R3



<https://www.sdx.com/>

Project: **HQLAx** - HQLAx

Blockchain Platform: **Corda (by R3)**

Token Standards / Types: Digital collateral tokens

Purpose: **Collateral mobility in finance**

Developer: Partners: Deutsche Börse, R3



<https://www.hqla-x.com/>

Project: **RedSwan CRE** - RedSwan.io

Blockchain Platform: **Stellar**

Token Standards / Types: Custom Stellar tokens

Purpose: **Commercial real estate tokenization**

Developer: Uses Stellar for \$100M+ in tokenized assets



<https://redswan.io/>

Project: **Ripple / sgBENJI** - DBS Bank + Franklin Templeton

Blockchain Platform: **Ripple (XRP Ledger)**

Token Standards / Types: Tokenized Money Market Fund (MMF)

Purpose: **Trading, yield generation via tokenized fund**

Developer: Developed with Ripple Labs



<https://digitalassets.frankltempleton.com/benji/>

Project: **Ripple Stablecoin** - Ripple Labs

Blockchain Platform: **Ripple (XRP Ledger)**

Token Standards / Types: RLUSD (Ripple stablecoin)

Purpose: **Enterprise-grade stablecoin for payments & liquidity**

Developer: Ripple Labs; launching RLUSD in 2025



<https://ripple.com/solutions/stablecoin/>

Project: **Agora DCM** - Agora Digital Capital Markets

Blockchain Platform: **Corda**

Token Standards / Types: Digital bond tokens

Purpose: **Syndication & issuance of bonds**

Developer: Corda-based issuance infrastructure



<https://agoradcm.com/>

Project: **Circle / USDC** - Circle - Cross-Chain Transfer Protocol (CCTP)

Blockchain Platform: **Ethereum, Solana, Stellar, others**

Token Standards / Types: ERC-20, SPL (on Solana), Stellar tokens

Purpose: **Stablecoin used in finance & tokenization**

Developer: Circle; backed by Coinbase and major VCs



<https://developers.circle.com/stablecoins/what-is-usdc>

Project: Tether / USDT - Tether Ltd.

Blockchain Platform: Tron, Ethereum, Solana, others

Token Standards / Types: TRC-20 (on Tron), ERC-20, SPL

Purpose: Stablecoins across multiple chains

Developer: Tether Ltd.; widely used in global remittances



<https://tether.to/ru/how-it-works/>

Project: Paxos - Paxos Trust Company

Blockchain Platform: Ethereum, Paxos Chain

Token Standards / Types: ERC-20 (PAX, PYUSD), regulated tokens

Purpose: Stablecoins, tokenized settlement assets

Developer: Regulated entity; partner of PayPal and banks



<https://www.paxos.com/>

Ethereum Token Standards

	Fungible	Non Fungible	Batch Transfer	Metadata Support	Native Platform
ERC 20	YES	NO	NO	LIMITED	ETHEREUM
ERC 721	NO	YES	NO	YES	ETHEREUM
ERC 1155	YES	YES	YES	YES	ETHEREUM
ERC 777 HOOKS	YES	NO	NO	YES	ETHEREUM
ERC 4337	N/A	N/A	N/A	N/A	ETHEREUM
ERC 6551	NO	EXTENDS	NO	Via ERC 721	ETHEREUM
ERC 4626	YES	NO	NO	OPTIONAL	ETHEREUM
ERC 2981	NO	METADATA	NO	ROYALTY	ETHEREUM

ERC 20 - Fungible Token Standard

Use case: Currencies, governance tokens, utility tokens. Fungibility:  Yes (every token is identical)

Core Functions (Smart Contract)

totalSupply() → Total tokens in circulation

balanceOf(address) → Balance of a given account

transfer(address to, uint256 amount) → Transfer tokens

approve(address spender, uint256 amount) → Approve third party to spend tokens

transferFrom(address from, address to, uint256 amount) → Move tokens on behalf of owner

allowance(address owner, address spender) → Check approved amount

Key Features

Simple, widely adopted, Doesn't natively support metadata or unique identifiers,

No built-in token recovery (transfers are final)

Architecture -

A single smart contract deployed per token. All balances and allowances are stored in mappings:

mapping(address => uint256) balances;

mapping(address => mapping(address => uint256)) allowances;

ERC 721 - Non Fungible Token Standard

Use case: Art, collectibles, game assets, real estate. Fungibility:  No (each token is unique)

Core Functions (Smart Contract)

`balanceOf(address owner), ownerOf(uint256 tokenId), transferFrom(address from, address to, uint256 tokenId), approve(address to, uint256 tokenId), getApproved(uint256 tokenId), setApprovalForAll(address operator, bool approved)`

Extensions

ERC 721 Metadata: For name, symbol, and token URI

ERC 721 Enumerable: To list all tokens owned by an address

Key Features

Unique token IDs represent unique assets, Metadata URI points to asset data (e.g., image, title), Typically uses IPFS or centralized storage for metadata. **Digital Ownership Becomes Real**

Architecture -

Token ownership stored in a mapping:

```
mapping(uint256 => address) private _owners;  
mapping(address => uint256) private _balances;
```

Each tokenId is globally unique.

ERC 1155 - Multi Token Engine Standard

Use case: Gaming, metaverse, hybrid assets (fungible + NFTs)

Fungibility: Both & (can handle fungible and non-fungible tokens)

Core Functions

`balanceOf(address, uint256 id),`
`balanceOfBatch(address[], uint256[]),`
`safeTransferFrom(address, address, uint256 id, uint256 amount, bytes data),`
`safeBatchTransferFrom(...)`

Features

Batch transfers of multiple tokens

Gas-efficient vs ERC-721 for NFT-heavy applications

Supports semi-fungible use cases (e.g., event tickets with editions)

Metadata via URI with ID substitution ({id})

Architecture -

Uses a **single contract to manage multiple token types**. Perfect for **gaming, collectibles, and economies that never sleep**. Tokens are identified by `uint256` IDs that can point to JSON metadata. Balances stored as a nested mapping: `mapping(uint256 => mapping(address => uint256)) private _balances;`

`id` distinguishes between different token types (ERC-20-like or NFT-like)

ERC 777 - Advanced Fungible Token Standard

Use Case: Currencies, DeFi with richer interfaces

Fungibility:  Yes

Core Functions

send(address to, uint256 amount, bytes data) → Transfer with data

authorizeOperator(address operator) → Authorize operator

revokeOperator(address operator)

isOperatorFor(address operator, address tokenHolder)

operatorSend(...), operatorBurn(...) → Operators can act on behalf of token holder

Key Features

Backward compatible with ERC-20

Supports hooks via ERC-1820 registry (more secure than “approve”)

Allows **sending data with tokens** (useful in DApps)

No front-running issue with “approve()” (unlike ERC-20)

Architecture

Uses the ERC-1820 registry to discover interface implementations

Operator model replaces “approve/transferFrom”

ERC 4337 - Account Abstraction via EntryPoint

Use case: Smart accounts (wallets with logic), gasless transactions, social recovery, paymaster integrations Fungibility:  (Not a token standard; it's an account abstraction layer)

Core Components

EntryPoint contract, UserOperation struct, handleOps(UserOperation[] ops, address beneficiary),
validateUserOp(UserOperation op, bytes32 userOpHash, uint256 missingFunds)

Features

Enables smart contract wallets to act like EOAs (Externally Owned Accounts)

No need for msg.sender or nonce tied to EOAs

Supports sponsored (gasless) transactions via Paymasters

Enables social recovery, multi-sig, session keys, and custom auth logic

Off-chain bundlers collect UserOperations and send them to the EntryPoint

Architecture

Introduces a new flow where users sign UserOperation structs instead of sending txs

Bundlers act like miners for smart wallets, submitting batched operations

The EntryPoint contract validates and executes UserOperations

Modular design allows for pluggable verification, gas payment strategies, etc.

Encourages innovation in wallet UX (biometrics, passkeys, plugins)

ERC 6551 - Token Bound Accounts (TBA)

Use case: **NFT identity, NFT wallets, dynamic NFTs** (NFTs that own other assets)

Fungibility:  **(Enhances NFTs with account capabilities)**

Core Functions

`createAccount(address implementation, uint256 chainId, address tokenContract, uint256 tokenId, uint256 salt, bytes initData)`

`account(address implementation, uint256 chainId, address tokenContract, uint256 tokenId, uint256 salt)`

Features

Allows NFTs to own other assets (tokens, NFTs, ETH)

NFT as an identity or profile with full smart contract wallet capabilities

Works with existing ERC-721s without modifying them

Each NFT gets a unique smart wallet (ERC-4337 compatible)

Enables composability: NFTs can now hold DeFi positions, memberships, or assets

Architecture

Each NFT can deterministically create a Token Bound Account (TBA) — a smart wallet

TBAs are deployed via a proxy factory pattern using create2

TBAs are tied 1:1 to an NFT by (contract, tokenId)

Only the current owner of the NFT controls the wallet

Ownership transfer of NFT = control transfer of the wallet

ERC 4626 – Tokenized Vault Standard

Use case: Yield-bearing vaults, DeFi strategies, interest-bearing tokens

Fungibility:  (built on ERC-20, fully fungible)

Core Functions

totalAssets(): Total underlying assets managed by the vault

convertToShares(uint256 assets): View how many vault shares would be minted for given assets

convertToAssets(uint256 shares): View how many assets are redeemable from given shares

deposit(uint256 assets, address receiver): Deposit assets and receive shares

withdraw(uint256 assets, address receiver, address owner): Withdraw assets by burning shares

mint(uint256 shares, address receiver): Mint exact number of shares by depositing necessary assets

redeem(uint256 shares, address receiver, address owner): Redeem shares for underlying assets

Features

Standard interface for vaults wrapping ERC-20 tokens

Enables interoperability between DeFi protocols

Supports automated yield strategies

(e.g., Yearn, Aave vaults)

Abstracts complex yield mechanics behind a clean API

Decouples asset management logic from frontend apps and integrators

Architecture

Built on ERC-20 standard — vault shares are fungible tokens

Vault holds and manages underlying ERC-20 assets

Internal accounting tracks proportional ownership via shares

Useful for abstracting lending, staking, and interest-bearing protocols

Encourages DeFi composability — vaults can be layered (e.g., vault-of-a-vault)

ERC 2981 – NFT Royalty Standard

Use case: Creator royalties, NFT marketplaces, secondary sales

Fungibility:  (used with ERC-721 or ERC-1155 NFTs)

Core Function

`royaltyInfo(uint256 tokenId, uint256 salePrice) → (address receiver, uint256 royaltyAmount)`

Returns the royalty payment address and amount owed based on the sale price.

Features

Standard method for querying royalty data across NFT marketplaces

Supports on-chain royalty enforcement or off-chain calculation

Royalty info is read-only; doesn't handle payments directly

Marketplace compatibility includes OpenSea, Rarible, etc.

Can be applied per-token or at the contract-wide level

Returns royalty receiver and amount as a percentage of sale

Doesn't interfere with transfer or ownership logic

Designed to enable fair monetization of creative work in NFTs

Integrators call `royaltyInfo()` during a sale to determine what royalty to pay

Non Ethereum Token Standards

	Fungible	Non Fungible	Batch Transfer	Metadata Support	Native Platform
BEP 20	YES	NO	NO	SAME AS ERC 20	BNB SMART CHAIN
TRC 20	YES	NO	NO	LIMITED	TRON
SPL Token	YES	NO (Metaplex)	NO	External	SOLANA
XLS 20 - XRP Token	YES	YES	Manual / via Hooks	YES	XRP LEDGER
FA 1.2	YES	NO	NO	LIMITED	TEZOS
FA 2	YES	YES	YES	YES	TEZOS
Stellar Tokens	YES	NO	NO	External / Anchors	STELLAR
Cardano Native	YES	YES	YES	YES	CADANO

BEP 20 - Binance Smart Chain Token Standard

Use Case: **BSC tokens (similar to ERC-20), stablecoins, DeFi**

Fungibility:  Yes

Core Functions

totalSupply()

balanceOf()

transfer()

approve()

transferFrom()

allowance()

Key Features

Based on **ERC-20 but deployed on BSC**

Lower fees than Ethereum

Often used **for bridging Ethereum tokens**

Architecture

Same structure as **ERC-20 with BSC-specific metadata extensions**

TRC 20 - Tron Token Standard

Use Case: **Stablecoins (e.g., USDT on Tron), DeFi**

Fungibility:  Yes

Core Functions

totalSupply()

balanceOf()

transfer()

approve()

transferFrom()

allowance()

Implements similar interfaces: **totalSupply, transfer, approve**, etc.

Smart contracts **written in Solidity or Tron's Java wrapper**

Hosted on the **Tron Virtual Machine (TVM)**

Gas fees are extremely low → popular for **Tether (USDT)**

Key Features

Nearly identical to ERC-20, runs on Tron VM

Extremely low transaction fees

Popular for stablecoin transfers

Architecture

Smart contracts on Tron blockchain (Solidity-compatible)

Uses TVM (Tron Virtual Machine)

SPL (Solana Program Library Token)

Use Case: **Fungible tokens on Solana – wrapped assets, utility tokens**

Fungibility:  Yes (for fungible SPL tokens)

 Core Functions (via Solana Token Program)

`mint_to()` → Mint tokens

`transfer()` → Transfer between accounts

`approve()` → Grant delegated authority

`revoke()` → Remove authority

`burn()` → Destroy tokens

 Key Features

Very fast and low-cost transactions

Off-chain metadata using Metaplex

Token accounts are separate from wallets

 Architecture

Uses associated token accounts:

Each user/token pair has a separate account

Managed by Solana Token Program (not per-token contracts)

Used on Solana for **both fungible and NFT tokens**

Accounts are structured differently due to **Solana's account-based model**

Smart contracts = Programs, and token balances are **stored in separate token accounts**

Metadata often handled off-chain via Metaplex standards

FA 1.2 - Tezos Fungible Token Standard

Use Case: **Currencies, DeFi tokens on Tezos**

Fungibility:  Yes

 Core Functions

getBalance(owner)

transfer(from, to, value)

approve(spender, value)

getAllowance(owner, spender)

 Key Features

Simpler model, similar to ERC-20

Commonly used for **basic tokens on Tezos**

 Architecture

Michelson/Michelson-like contracts

Token ledger via big_maps

FA 2 - Tezos Multi Asset Token Standard

Use Case: **Fungible + Non-Fungible tokens (NFTs)**

Fungibility:  Both supported

 Core Functions

balance_of(requests)

transfer(list of transfers)

update_operators()

token_metadata()

 Key Features

Supports multiple token types in a single contract

Rich metadata standard

Operator-based transfers

 Architecture

Similar to ERC-1155

Token balances stored as nested maps:

big_map (token_id, owner) => balance

XLS 20 - XRP Ledger Token Standard

Use Case: **Issued Currencies (Fungible), NFTs via XLS-20**

Fungibility:  Both supported (via different standards)



XLS-20 – NFT standard

XLS-14d / XLS-15d – Decentralized identifiers, trust mechanisms

XRP Ledger Issued Currencies – Fungible token system built-in

Uses “trust lines” to represent token balances between parties

No smart contracts in the Ethereum sense — relies on built-in ledger logic

XLS-20 standard: Introduced native NFT support on the XRP Ledger
RLUSD (Ripple's stablecoin) and other tokens will follow native issuance logic



Native token issuance without smart contracts

No gas fees – uses XRP for transaction fees

Trust line mechanism to manage token risk

XLS-20 supports royalties, transfer hooks (NFTs)



Fungible tokens: Issued by accounts via IOU model

Account issues a token like USD.Tether or BTC.Bitstamp

NFTs (XLS-20):

Native NFT objects on-ledger

Metadata, royalties, and sell offers embedded

Trust Lines:

Account ↔ Token issuer connection required to hold tokens

balance = TrustLine object

Stellar Tokens (SEP Standards)

Use Case: **Fungible tokens (USD, BTC), Stablecoins, Asset-backed tokens**

Fungibility:  **Fungible only** (NFTs are not natively supported)

 Core Protocol

SEP-0005 / SEP-0010 / SEP-0024 – Federation, auth, KYC, token distribution

Issuers create assets with a code + issuer address (e.g., USD:GDUKMG...)

Operations:

`payment()`, `change_trust()`, `allow_trust()`, `create_offer()`

 Key Features

No smart contracts (logic via **protocol-level operations**)

Trust lines for managing token exposure

Supports compliance: flags for KYC, freeze, authorization

Anchor model for real-world asset issuance

Assets are **issued directly by an account (issuer)** — no smart contract

Stellar uses **asset pairs (issuer + asset code)** to define a token

Trustlines must be set by users to accept assets

Smart contract functionality is emerging via Soroban (Stellar's WASM-based smart contracts)



Architecture

Native Asset: XLM

Custom Assets: Code + Issuer

e.g., `Token("USD", issuer_address)`

Balances tracked per account via

TrustLines

`Account ↔ Asset (issuer) ↔ balance`

Cardano Native Tokens

Use Case: **Fungible and Non-Fungible tokens (NFTs)**

Fungibility:  Both supported



Minting/burning via Minting Policies

Transactions directly include native tokens

No smart contracts needed for basic token use



True native assets: Same treatment as ADA

No smart contract required to create/send tokens

NFTs = tokens with quantity = 1 + unique policy

Deterministic monetary policy via scripts



Multi-Asset Ledger:

Native support for custom assets at the ledger level

Token Format: <PolicyID>.<AssetName>

Minting Policy Script:

Defines how/when tokens can be minted or burned e.g., Time-locked, multisig

Balance Storage:

Account (address) holds bundle of assets:

{

ADA: 10.5,
policy1.TokenA: 1000,
policy2.NFT001: 1

}

Tokens are treated like first-class citizens (**no contract required**)

Minting and burning controlled via minting policy scripts

Smart contracts are **written in Plutus (Haskell-based)**

HOW TO GET TOKENS

*Obtain relevant wallets that support the desired token standard.

Participate in dApps, exchanges, or token sale events that use these token formats.

*For developers, build and deploy smart contracts adhering to the chosen token standards for custom assets and decentralized services.

*Token standards enable a thriving ecosystem by ensuring compatibility, security, and flexibility across blockchains, empowering developers and users with tools for innovation and interaction.