# Report:

# ERC-1155 & Fractional Ownership Use Cases - Case 1 Real Estate

**Prepared By:** Abhilash Rajan
**Date:** 27 October 2025

# 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 |

## Overview - ERC 1155 - Multi Token Engine Standard

- A **multi-token standard** for Ethereum, designed by Enjin.
- Allows a **single smart contract to manage multiple token types** — both fungible and non-fungible tokens.
- A **robust, efficient, and scalable choice** for fractional ownership applications in real estate or other asset classes.
- Perfect Use Case 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)

**Key Features:**

- **Multi-token efficiency: Mint, transfer, or burn** multiple token types in a single transaction.
- **Reduced gas costs: Batch operations** are cheaper than using multiple ERC-20 or ERC-721 contracts.
- **Flexible metadata:** Each token **ID can have unique metadata**.
- **Safe transfers:** Includes **safeTransferFrom and batch transfers** to **prevent tokens from being lost in contracts** that don't handle them.
- **Interoperability:** Supported by major **marketplaces like OpenSea, Enjin, and Zora**.

## Core Functions

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

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol";

contract MyERC1155 is ERC1155 {
    constructor() ERC1155("https://api.example.com/metadata/{id}.json") {}
    function mint(address to, uint256 id, uint256 amount, bytes memory data) public {
        _mint(to, id, amount, data);    } }
```

Compile - Deploy

Call mint with token ID, amount, and empty data (0x)

# Use Cases of ERC-1155

Widely used in applications that need **multiple token types in a single ecosystem**.

**Major Use Cases:**

**Gaming**: In-game items: swords, shields, consumables, etc.
Example: Enjin Coin, The Sandbox etc platforms use ERC-1155 to **represent virtual items**.

**NFT Collections**: **Unique Limited edition items** with varying supply.
Example: OpenSea, Rarible, Zora supports ERC-1155 NFT drops.

**Fractional Ownership**: **Real estate, art, or collectibles can be divided into shares** using ERC-1155. Owners can **trade, lease, or earn revenue** proportionally. Example: RealT, Lofty AI

**Ticketing Systems**: Event tickets of multiple types in one contract.
Example: Mintbase, POAP issuing **general, VIP, and backstage passes**.

**DeFi & Staking**: Represent **LP tokens, derivatives, or reward tokens** in a single contract.

**Commodities or Resource Tokens:** Represent **units of energy, raw materials, or products**.
Example: Energy Web, Supply-chain dApps

# Smart Contract Summary (ERC-1155)

To tokenize a building into multiple rooms (each room = 1 token ID). Each room has fractional shares representing co-ownership.

- Room Initialization: **_initializeBuilding()** rooms, assigns types, rents, and mints fractional tokens
- Fraction Creation: **_mint() (in _initializeBuilding())** Creates 100 ERC-1155 tokens per room representing fractional ownership
- Fractional Transfer:**transferOwnershipFraction(), safeTransferFrom** for secure token transfer, **OwnershipTransferredERC1155** to record the transfer on-chain.
- Leasing: **buyShares()** allows tenants to **pay rent and occupy a room**.
- Time-Bound Lease Tracking: Mapping stores the tenant's address, start block, and end block. **The checkLeaseStatus()** allows to check if the **lease is active or expired**.
- Past Tenants: **getRoom() / getTenants()** fetches metadata and tenant history for transparency
- Metadata: **uri(roomId)** points to **JSON metadata hosted on IPFS** for OpenSea or marketplaces.
- Security: **ReentrancyGuard prevents reentrancy attacks during ETH transfers**. **onlyOwner** ensures sensitive operations like **rent modification are restricted**.
- Deactivate Leasing: **setForLease()** to enables the owner to activate/deactivate leasing or adjust rent rates

## Functional Flow Summary

- **Each unit (room/shop) is tokenized as an ERC1155 asset.**

- **Ownership can be divided and traded as fractional shares.**

- **Rent and lease operations are automated via smart contracts.**

- **Revenue management is transparent and tamper-proof.**