ERC-1155 MULTI-TOKEN STANDARD

Student Name: Keshav Murthy Ramachandra
UB id: 50360333
Supervisor: Dr. Bina Ramamurthy

Introduction

What is a Multi-Token Standard?

The idea is to create a smart contract interface that can represent and control any number of fungible and non-fungible token types. In this way, the ERC-1155 token can do the same functions as an ERC-20 and ERC-721 token, and even both at the same time.

ERC-1155 is a standard interface for contracts that manage multiple token types. A single deployed contract may include any combination of fungible tokens, non-fungible tokens or other configurations such as semi-fungible tokens.

ERC-1155 FUNCTIONS AND FEATURES:

Batch Transfer: Transfer multiple assets in a single call.

The batch transfer works very similar to regular ERC-20 transfers. The only difference in ERC-1155 is that we pass the values as an array and we also pass an array of ids. For example given ids=[1, 5, 7] and values=[10, 20, 50], the resulting transfers will be

Transfer 10 tokens with id 1 from _from to _to.
Transfer 20 tokens with id 5 from _from to _to.
Transfer 50 tokens with id 7 from _from to _to.

```
function safeBatchTransferFrom(
    address _from,
    address _to,
    uint256[] calldata _ids,
    uint256[] calldata _values,
    bytes calldata _data
) external
```

Batch Balance: Get the balances of multiple assets in a single call.

we can retrieve multiple balances in a single call. We pass an array of owners, followed by the array of token ids.

```
function balanceOfBatch(
    address[] calldata _owners,
    uint256[] calldata _ids
) external view returns (uint256[] memory);
```

Batch Approval: Approve all tokens to an address.

```
function setApprovalForAll(
    address _operator,
    bool _approved
) external;

function isApprovedForAll(
    address _owner,
    address _operator
) external view returns (bool);
```

Unlike in ERC-20, Instead of approving specific amounts, you set an operator to be approved or not approved using setApprovalForAll.

Reading the current status can be done via isApprovedForAll. As you can see, it's all or nothing. You cannot define how many tokens to approve or even which token class.

Hooks: Receive tokens hook.

ERC-1155 supports receive hooks for smart contracts only.

The hook function must return a magic predefined bytes4 value as below:

bytes4(keccak256("onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)"))

When the receiving contract returns this value, it is assumed the contract accepts the transfer and knows how to handle the ERC-1155 tokens. This facilitates for tokens not getting stuck in the Smart Contract.

```
function onERC1155BatchReceived(
    address _operator,
    address _from,
    uint256[] calldata _ids,
    uint256[] calldata _values,
    bytes calldata _data
) external returns(bytes4);
```

NFT Support: If supply is only 1, treat it as NFT.

When the supply is just one, the token is essentially a non-fungible token (NFT). And as is standard for ERC-721, we can define a metadata URL. The URL can be read and modified by clients.

Safe Transfer Rules: Set of rules for secure transfer.

Examples :

- The caller must be approved to spend the tokens for the _from address or the caller must equal _from.
- The transfer call must revert if
    1. _to address is 0.
    2. length of _ids is not the same as length of _values.
    3. any of the balances of the holders for tokens in _ids is lower than the respective amounts in _values sent to the recipient.
    4. any other error occurs.

Advantages of ERC-1155 ?

Multiple Tokens : We can define and configure both fungibles and NFTs in a single smart contract.
Save Gas : We can cut gas fees by up to 90% when minting new tokens.
Advanced Features : Enable users to do everything from destroying to upgrading NFTs.
Atomic Swaps : Enable atomic swaps of any amount of tokens in just two simple steps.
Batch Transfers : Send multiple tokens in a single transaction.

ERC-721 vs. ERC-1155

- ERC-1155 permits the creation of both semi-fungible tokens and non-fungible tokens, whereas ERC-721 permits only the latter.
- In ERC-1155, smart contracts are linked to multiple URIs and do not store additional metadata. In comparison, ERC-721 only supports static metadata stored directly on the smart contract for each token ID, increasing deployment costs and limiting flexibility.
- ERC-1155's smart contracts support an infinite number of tokens, whereas ERC-721 needs a new smart contract for each type of token.
- ERC-1155 also allows batch transfers of tokens, which can reduce transaction costs and times. With ERC-721, if you want to send multiple tokens, they happen individually.

How to create a ERC 1155 Smart Contract

In the Example, we'll use ERC1155 to track multiple items in a game, each one of them having their own unique attributes. We mint all items to the deployer of the contract, which we can later transfer to players. Players are free to keep their tokens or trade them with other people.

```
// contracts/GameItems.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

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

contract GameItems is ERC1155 {
    uint256 public constant GOLD = 0;
    uint256 public constant SILVER = 1;
    uint256 public constant THORS_HAMMER = 2;
    uint256 public constant SWORD = 3;
    uint256 public constant SHIELD = 4;

    constructor() public ERC1155("https://game.example/api/item/{id}.json") {
        _mint(msg.sender, GOLD, 10**18, "");
        _mint(msg.sender, SILVER, 10**27, "");
        _mint(msg.sender, THORS_HAMMER, 1, "");
        _mint(msg.sender, SWORD, 10**9, "");
        _mint(msg.sender, SHIELD, 10**9, "");
    }
}
```

Note that for our Game Items, Gold is a fungible token whilst Thor's Hammer is a non-fungible token as we minted only one.

Once deployed, we will be able to query the deployer's balance

```
> gameItems.balanceOf(deployerAddress,3)
1000000000
```

We can transfer items to player accounts:

```
> gameItems.safeTransferFrom(deployerAddress, playerAddress, 2, 1, "0x0")
> gameItems.balanceOf(playerAddress, 2)
1
> gameItems.balanceOf(deployerAddress, 2)
0
```

The metadata uri can be obtained:

```
> gameItems.uri(2)
"https://game.example/api/item/{id}.json"
```

The uri can include the string {id} which clients must replace with the actual token ID, in lowercase hexadecimal (with no 0x prefix) and leading zero padded to 64 hex characters.

Conclusion:

The ERC1155 exhibits features of both the ERC 20 and ERC 721 thus providing accessibility, simplicity, and efficiency on the buyer's side.