---

title: ERC-1155 Multi-Token Standard description: lang: en

Introduction {#introduction}

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 (e.g. semi-fungible tokens).

What is meant by Multi-Token Standard?

The idea is simple and seeks 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. It improves the functionality of both the ERC-20 and ERC-721 standards, making it more efficient and correcting obvious implementation errors.

The ERC-1155 token is described fully inEIP-1155.

Prerequisites {#prerequisites}

To better understand this page, we recommend you first read abouttoken standards, ERC-20, and ERC-721.

ERC-1155 Functions and Features: {#body}

- Batch Transfer: Transfer multiple assets in a single call.
- Batch Balance: Get the balances of multiple assets in a single call.
- Batch Approval: Approve all tokens to an address.
- Hooks: Receive tokens hook.
- NFT Support: If supply is only 1, treat it as NFT.
- Safe Transfer Rules: Set of rules for secure transfer.

Batch Transfers {#batch-transfers}

The batch transfer works very similar to regular ERC-20 transfers. Let's look at the regular ERC-20`transferFrom` function:

```solidity // ERC-20 function transferFrom(address from, address to, uint256 value) external returns (bool);

// ERC-1155 function safeBatchTransferFrom( address _from, address _to, uint256[] calldata _ids, uint256[] calldata _values, bytes calldata _data ) external; ```

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=[3, 6, 13]` and `values=[100, 200, 5]`, the resulting transfers will be

1. Transfer 100 tokens with id 3 from`_from` to`_to`.
2. Transfer 200 tokens with id 6 from`_from` to`_to`.
3. Transfer 5 tokens with id 13 from`_from` to`_to`.

In ERC-1155 we only have`transferFrom`, no`transfer`. To use it like a regular`transfer`, just set the from address to the address that's calling the function.

## Batch Balance {#batch-balance}

The respective ERC-20`balanceOf` call likewise has its partner function with batch support. As a reminder, this is the ERC-20 version:

```solidity // ERC-20 function balanceOf(address owner) external view returns (uint256);

// ERC-1155 function balanceOfBatch( address[] calldata _owners, uint256[] calldata _ids ) external view returns (uint256[]

memory); ```

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

For example given `_ids=[3, 6, 13]` and `_owners=[0xbeef..., 0x1337..., 0x1111...]`, the return value will be

`solidity [ balanceOf(0xbeef...), balanceOf(0x1337...), balanceOf(0x1111...) ]`

### Batch Approval {#batch-approval}

```solidity // ERC-1155 function setApprovalForAll( address _operator, bool _approved ) external;

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

The approvals are slightly different than ERC-20. Instead of approving specific amounts, you set an operator to approved or not approved via `setApprovalForAll`.

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

This is intentionally designed with simplicity in mind. You can only approve everything for one address.

### Receive Hook {#receive-hook}

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

Given the EIP-165 support, ERC-1155 supports receive hooks for smart contracts only. The hook function must return a magic predefined bytes4 value which is given as:

`solidity 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. Great, no more stuck tokens in a contract!

### NFT Support {#nft-support}

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

### Safe Transfer Rule {#safe-transfer-rule}

We've touched on a few safe transfer rules already in the previous explanations. But let's look at the most important of the rules:

1. The caller must be approved to spend the tokens for the `_from` address or the caller must equal `_from`.
2. The transfer call must revert if
3. `_to` address is 0.
4. length of `_ids` is not the same as length of `_values`.
5. any of the balance(s) of the holder(s) for token(s) in `_ids` is lower than the respective amount(s) in `_values` sent to the recipient.
6. any other error occurs.

*Note*: All batch functions including the hook also exist as versions without batch. This is done for gas efficiency, considering transferring just one asset will likely still be the most commonly used way. We've left them out for simplicity in the explanations, including safe transfer rules. The names are identical, just remove the 'Batch'.

## Further reading {#further-reading}

- EIP-1155: Multi Token Standard
```

- ERC-1155: Openzeppelin Docs
- ERC-1155: GitHub Repo
- Alchemy NFT API