BLOCK WORLD

catalogue

0x0 Project background

This technical white paper explains some of the design logic and economic models behind the Block World core contract. It's being used to solve the entire multi-chain ecology. After Web3, humans will enter a whole new digital age. Block World will be the digital currency that dominates the economy, and its gaming technology and smart contracts bind to a new revolutionary approach that allows every CPU to participate. As technology continues to advance, traditional cryptocurrency games are becoming more centralized, and only miners with a lot of dedicated hardware and huge computing power can benefit. However, Block World's gaming technology has completely changed that.

The Block World game technology is based on a brand new algorithm that takes advantage of the special properties of the Block World structure. Each CPU can be bound to the Block World smart contract and participate in the mining process of virtual coins.

To ensure fairness and security, Block World's games requires smart contracts. A smart contract is an automatically executed computer program that contains the rules and conditions of the game. To protect users' privacy and data security.

0x1 Preface

"Bitcoin" is the successful implementation of the p2p e-cash concept. Both professionals and the general public are beginning to appreciate the convenient combination of public transactions and work certificates as a trust model. Today, the user base of e-cash is growing steadily growing; consumers are attracted by low fees and the anonymity of e-cash provision, and merchants value their projected and scattered emissions. Bitcoin effectively demonstrates that e-cash can be as simple as paper money and as convenient as a credit card.

However, Bitcoin does not complete the smart contracts like Ethereum, as a smart contract can complete a lot of blockchain functions. Block World Built on the Ethereum smart contract, it can solve the incompatible contradictions in the blockchain game through the ecology. It can be both inflated and deflationary. Through GAMEFI, the NFT ecosystem is destroyed to achieve deflation, inflation is achieved through the node system, the deflation part is destroyed, and the inflation part is released to the node.

Digital assets and scarcity: sci-fi blockchain games typically issue digital assets such as virtual planets, space fleets, alien creatures, etc. These assets are unique and scarce on the blockchain, and players can actually own them, and they also have value outside of the game world. Decentralized world: sci-fi blockchain games tend to build decentralized virtual worlds, similar to "metacogies" where players are free to explore, build and interact, which are subject to the rules and constraints of smart contracts. Autonomy and Control: Players often have more autonomy and control in science fiction blockchain games, where they can freely trade, sell, lease or otherwise manage their digital assets, all transparent. Interoperability: Some sci-fi blockchain games aim to achieve interoperability across platforms and across games. That means you can migrate your virtual space fleet from game to game, or use your digital alien pet in multiple games. Economic systems: Science fiction blockchain games often have complex economic systems, including tokens, currencies, and markets, that can encourage players to participate in the game world and create value. Real world value: Digital assets in sci-fi blockchain games can have real world value because they can be traded inside and outside the game. This allows players to make gains or investments in the game.

In general, science fiction blockchain games combine the creativity of science fiction with the transparency and decentralization of blockchain technology, providing players with more freedom and control, while also creating a virtual world with real value.

0x 2 Major issues in the current blockchain ecology

0x20 classic Token price determinants

The price of a token is determined by several factors, including:

Supply and demand: The basic economic principles of supply and demand are the main factor determining the price of a token. If demand for tokens is high and supply is limited, prices could rise. If demand is low and supply is high, prices may fall.

Market Sentiment: The market's overall perception and attitude towards tokens will also have a significant impact on their prices. Positive news and developments can increase sentiment and push up prices, while negative news and events can lower sentiment and lower prices.

Adoption and use: The wider the adoption and use of tokens, the higher the value may be. Tokens with strong use cases, a large active user base, and strong partnerships are more likely to add value over time.

Competition: The presence of similar or competing tokens also affects the price of the token. A new tokens with similar functions and use cases may reduce the demand for the original tokens, resulting in a lower price.

Regulatory environment: The regulatory environment can also play a role in determining the price of tokens. If the token is subject to restrictive regulations, it may reduce demand and lower prices, while favorable regulations may increase demand and raise prices.

These are some of the key factors that may affect the price of a token, the cryptocurrency market is highly volatile and unpredictable, and token prices may be influenced by many other factors.

Tokens weigh the supply and demand relationship of typical models based on tokens, and consider the influence of domestic factors (such as blockchain financial growth rate, inflation rate, interest rate, etc.) and the overall market value factors.

0x 21 Classic inflation-price incompatibility paradox

In classical economics, moderate inflation can stimulate the economy, but excessive inflation can cause violent fluctuations in the capital market and cause chaos in the capital market. According to the market rules, the greater the inflation, the lower the price. This has been fully reflected in the digital currency market. The total inflation of many coins cannot be controlled, leading to a lower price spiral.

Block World Solve the classic inflation value incompatibility paradox through smart contracts. More algorithms can solve the problem of inflationary and deflation. The inflation part is given the weight to the node user. Through the combination of game output and NFT, the whole system locks in liquidity, reduces the positive correlation between inflation and price, makes inflation compatible with price, and forms a closed loop of ecological economy.

0x22 Block World Price algorithm scheme

Algorithmic price formula refers to the mathematical equation used to determine how the Token price should be established as conditions or other factors change. Depending on the assets assessed and the method of analysis, different models may be used.

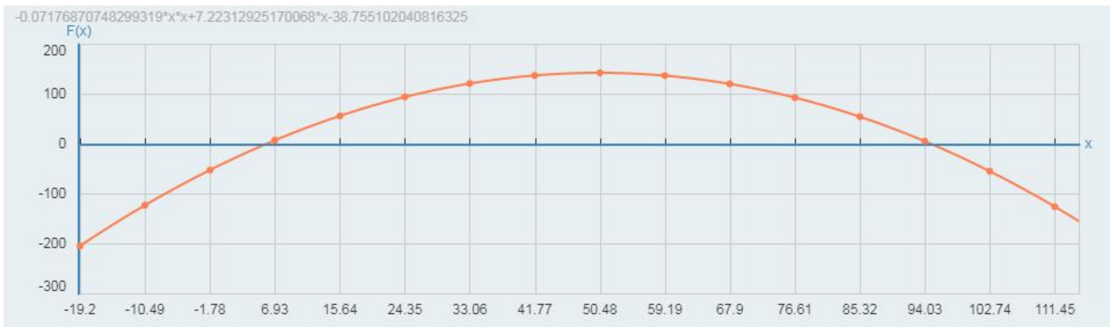
Block World Is a service as a platform. In order to enable many projects to have an ecosystem, it provides project client services, which are destroyed or produced through contracts. The destruction of destruction permanently into the black hole address address (0). Block World The contract releases the same proportion of Token within a certain period of time according to the player's situation, which is issued as the player's equity certificate.

0x23, the underlying formula

Token, Total Output Calculation:

$$Harvest = \sum_{n=0}^{burn} = Gamefi.burn + 2Gamefi.burn + \ldots\ldots + mGamefi.burn$$

Fit the Token total amount curve:



X standard: total output

Y standard: total flow volume

Token combustion calculation:

$$Burn = \sum_{n=0}^{Game.burn} = Master.burn + Attack.burn + \ldots\ldots + Set.burn$$

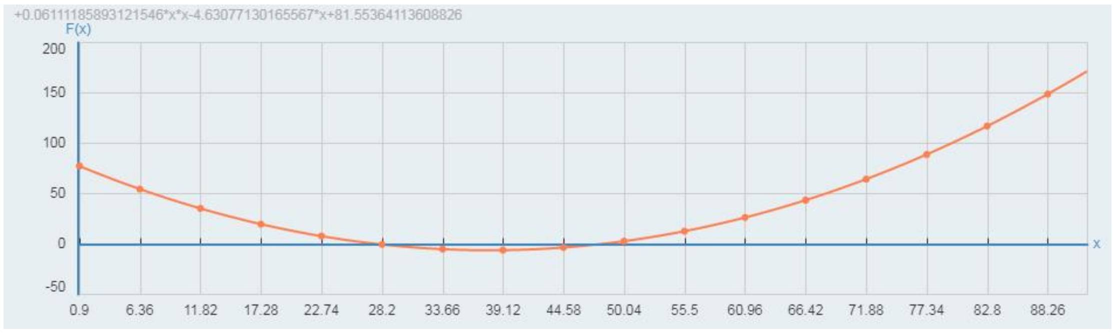
Token Price curve

$$Price^{(n)} = Total \times Nft.price/(Total - Burn)$$

Fit the price curve:

X mark: Nft market value

Y standard: Token value



0x3 game reward ecology

. 0x30 game world ecology

Genesis players: Block World Genesis players are created from when the project is started. When everyone enters the game, it is fair.

Player revenue formula:

$$Bonus = \sum_0^n SelfRewards + Plunder$$

0x31 GameFi

Combined with Game (Game) Finance (Finance) Mining (Games)

GameFi The ecosystem uses cryptocurrencies, non-homogeneous tokens (NFT), and blockchain technology to create virtual games.

Block World Ecological games consof the following subjects.

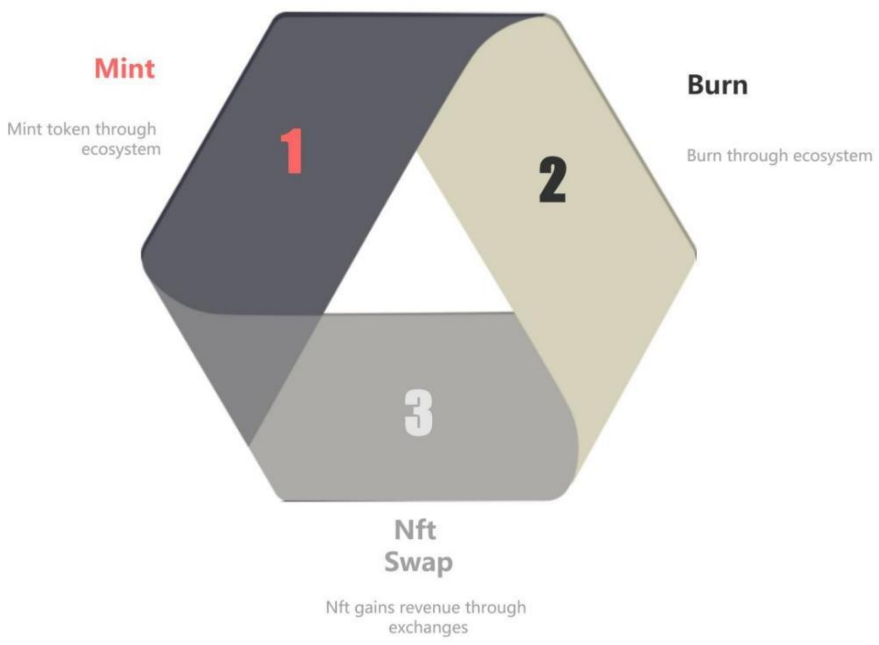
Users: Play and trade virtual assets in games using the GameFi ecosystem. Make money in the play.

Token: The Block World ecosystem uses full chain mode and can be used as virtual assets in games and other ecological applications.

Smart game contracts: smart contracts used in the ecosystem to support game trading and economic models. Let the game destruction and game additional issuance to achieve a certain balance.

Architecture platform: a developer platform that provides technical support for the ecosystem and develops new features.

Through these elements, the Block World ecosystem tries to solve the incompatible economic model and bring the results combined with smart contracts into the gaming space, providing a stable, transparent, and credible closed loop of economic gaming.


Mint
Mint token through ecosystem
1
Burn
Burn through ecosystem
2
3
Nft Swap
Nft gains revenue through exchanges

0x32 Block World Financial ecology

It mainly consists of three parts:

Coin casting: release the income of node users through coin casting, and coin casting is actually increasing the circulation part.

Combustion: Burning off the Token through the ecosystem. The amount of combustion can be calculated and adjusted according to the ecological situation.

NFT economy: Buy NFT through the game player configuration, and buy back part of the TOKEN.

Through these three parts, the economic closed loop of the whole ecology of Block World is formed to avoid malicious inflation and excessive deflation.

0x33Block World War Games

Block World War game ecology only provides the basic consumption as a game platform, while the game itself requires other project parties to enter the whole ecology. There are two main tokens in the platform, respectively Block World Game and Block World NFT.

Token (n) (Token of the participating ecological project party)

Aggression / defense force

Block World To add the game base attributes token

1-point energy = 1 token

1 token = 1 point of attack or defense

Entering the Block World war system, each adventurer is sheltered by kingdom energy, adding different country tokens depending on the country the player is in.

scene

1. Gameplay

There are many countries on the magical crypto continent. As a warrior of the country, players can summon the red dragon, Thai tower, archangel, nine-headed dragon and other summoning beasts for adventure. Players can play the role of a castle master, building a powerful force through expeditions around the world. Players can also go to other countries, communicate with different forces, bring down all kinds of powerful enemies, and build an immortal legend by winning an epic war.

2. Income from Token (n)

User revenue is determined based on the user attack power and the country base and NFT bonus,

Output country token = (attack force / (country base) Xnft bonus) * min

If the attack force is greater than the defense of the other side, you can rob the other side did not get the income token

3. Calculation formula of deflation T-H attenuation:

$$T\frac{\mathrm{d}H}{\mathrm{d}t}+H=2H_0$$

$$T_1T_2\frac{\mathrm{d}^2H_2}{\mathrm{d}t^2}+(T_1+T_2)\frac{\mathrm{d}H_2}{\mathrm{d}t}+H_2=rk_\mu\mu R_1$$

$$T_1T_2'T_3\frac{\mathrm{d}^3H_3}{\mathrm{d}t^3}+(T_1T_2+T_1T_3+T_2T_3)\frac{\mathrm{d}^2H_3}{\mathrm{d}t^2}+$$
$$(T_1+T_2+T_3)\frac{\mathrm{d}H_3}{\mathrm{d}t}+H_3=r_1r_2k_\mu\mu R_1$$

4. Globalized ecological king

In order to defeat other countries, each country produces a king, and the king earns twice as much as the other players, and wins the title by challenging the king to defeat the king.

5. Game inflation formula:

$$\Delta Reward=\begin{cases}0 & i_c<i_l\\ \Delta F\cdot\left(\sqrt{N}-\sqrt{N(i_l)}\right) & i_l\le i_c<i_u\\ \Delta F\cdot\left(\sqrt{N(N_u)}-\sqrt{N(i_l)}\right) & i_c\ge i_u\end{cases}$$

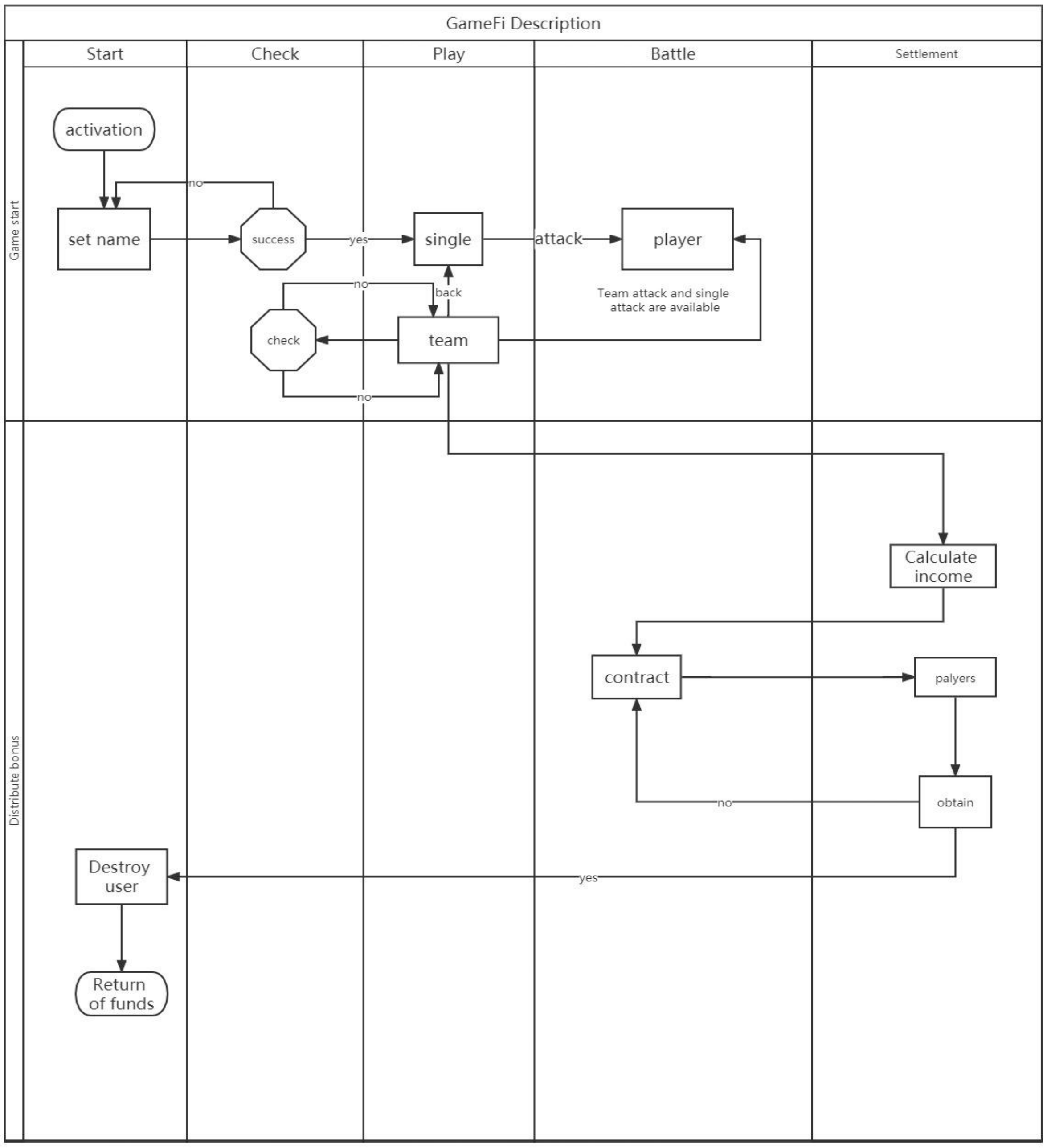
$$\Delta Reward=\begin{cases}\Delta G\cdot\left(\frac{1}{\sqrt{M(i_l)}}-\frac{1}{\sqrt{M(i_u)}}\right) & i_c<i_l\\ \Delta G\cdot\left(\frac{1}{\sqrt{M}}-\frac{1}{\sqrt{M(i_u)}}\right) & i_l\le i_c<i_u\\ 0 & i_c\ge i_u\end{cases}$$

0x34 NFT

Block World The limited version of the War NFT is the ancillary property of the game.

1. Users can go to the NFT exchange to buy NFT items to improve the player's attributes.
2. The NFT equipment of Block World War is limited edition, adding three ordinary attributes and two special attributes. The attack force defense was increased by percentage, and the attack cooldown was reduced by N seconds. The special attribute is remote attack, time travel.

Game flow chart



The NFT endorses the Block World economic ecology. NFT sales are all the main coins on the chain. After sales, Block World tokens are regularly repurchased to solve the problem of ecological price inflation conflict.

NFT parameter list

Name	Num	PRICE	ATT	DEF	TIME	STUNT
VALKYRIE	400	2	2	0	0	NULL
ENRAGED CYCLOPIA	400	2	1	2	0	NULL
ANCIENT TREANT	100	4	2	6	0	NULL
ABBOT	100	4	0	4	200	CHAT
GRIM REAPER	100	10	13	2	0	NULL
EMERALD DRAGON	50	20	10	10	1500	CHAT
SIMURGH	50	20	20	5	0	DISTANCE SPACE
SPECTRAL DRAGON	50	20	10	35	100	SPACE
ANCIENT BEHEMOTH	50	20	30	1	100	DISTANCE
DARK HYDRA	50	20	20	20	100	SPACE
TITAN	50	20	10	40	100	SPACE
ARDENT DRAGON	30	30	40	20	100	DISTANCE SPACE
CELESTIAL	20	50	50	50	400	ALL
BLACK DRAGON	20	50	60	30	400	ALL

0x4 Project timeline

Q 2,2023: Complete the feasibility demonstration of the economic model of the Block World project.

Q 32023: Complete the game core algorithm simulation.

Q 22023: Complete the GamefiBlock World Wargame Core Smart contract code.

Third quarter of 2023: Block World DAPP Completed.

Q 42023: Block World Game Launch.

Q 1,2024:.............................

0x5 contract code

The NFT contract code

```
// File: openzeppelin-contracts-master/contracts/introspection/IERC165.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.6.2;

/**
* @dev Interface of the ERC165 standard, as defined in the
* https://eips.ethereum.org/EIPS/eip-165[EIP].
*
* Implementers can declare support of contract interfaces, which can then be
* queried by others ({ERC165Checker}).
*
* For an implementation, see {ERC165}.
*/
interface IERC165 {
/**
* @dev Returns true if this contract implements the interface defined by
* `interfaceId`.See the corresponding
* https://eips.ethereum.org/EIPS/eip-165#how-interfaces-are-identified[EIP section]
* to learn more about how these ids are created.
*
* This function call must use less than 30 000 gas.
*/
function supportsInterface(bytes4 interfaceId) external view returns (bool);
}

// File: openzeppelin-contracts-master/contracts/token/ERC1155/IERC1155.sol

pragma solidity ^0.6.2;

/**
* @dev Required interface of an ERC1155 compliant contract, as defined in the
* https://eips.ethereum.org/EIPS/eip-1155[EIP].
*
* _Available since v3.1._
*/
interface IERC1155 is IERC165 {
/**
```

```
* @dev Emitted when `value` tokens of token type `id` are transferred from `from` to
`to` by `operator`.
*/
event TransferSingle(address indexed operator, address indexed from, address
indexed to, uint256 id, uint256 value);

/**
* @dev Equivalent to multiple {TransferSingle} events, where `operator`, `from` and
`to` are the same for all
* transfers.
*/
event TransferBatch(address indexed operator, address indexed from, address
indexed to, uint256[] ids, uint256[] values);

/**
* @dev Emitted when `account` grants or revokes permission to `operator` to
transfer their tokens, according to
* `approved`.
*/
event ApprovalForAll(address indexed account, address indexed operator, bool
approved);

/**
* @dev Emitted when the URI for token type `id` changes to `value`, if it is a
non-programmatic URI.
*
* If an {URI} event was emitted for `id`, the standard
* https://eips.ethereum.org/EIPS/eip-1155#metadata-extensions[guarantees] that
`value` will equal the value
* returned by {IERC1155MetadataURI-uri}.
*/
event URI(string value, uint256 indexed id);

/**
* @dev Returns the amount of tokens of token type `id` owned by `account`.
*
* Requirements:
*
* - `account` cannot be the zero address.
*/
function balanceOf(address account, uint256 id) external view returns (uint256);

/**
* @dev xref:ROOT:erc1155.adoc#batch-operations[Batched] version of {balanceOf}.
```

```
*
* Requirements:
*
* - `accounts` and `ids` must have the same length.
*/
function balanceOfBatch(address[] calldata accounts, uint256[] calldata ids) external
view returns (uint256[] memory);

/**
* @dev Grants or revokes permission to `operator` to transfer the caller's tokens,
according to `approved`,
*
* Emits an {ApprovalForAll} event.
*
* Requirements:
*
* - `operator` cannot be the caller.
*/
function setApprovalForAll(address operator, bool approved) external;

/**
* @dev Returns true if `operator` is approved to transfer ``account``'s tokens.
*
* See {setApprovalForAll}.
*/
function isApprovedForAll(address account, address operator) external view returns
(bool);

/**
* @dev Transfers `amount` tokens of token type `id` from `from` to `to`.
*
* Emits a {TransferSingle} event.
*
* Requirements:
*
* - `to` cannot be the zero address.
* - If the caller is not `from`, it must be have been approved to spend ``from``'s
tokens via {setApprovalForAll}.
* - `from` must have a balance of tokens of type `id` of at least `amount`.
* - If `to` refers to a smart contract, it must implement
{IERC1155Receiver-onERC1155Received} and return the
* acc Block World ance magic value.
*/
function safeTransferFrom(address from, address to, uint256 id, uint256 amount,
bytes calldata data) external;

/**
* @dev xref:ROOT:erc1155.adoc#batch-operations[Batched] version of
{safeTransferFrom}.
*
* Emits a {TransferBatch} event.
*
* Requirements:
*
* - `ids` and `amounts` must have the same length.
* - If `to` refers to a smart contract, it must implement
{IERC1155Receiver-onERC1155BatchReceived} and return the
* acc Block World ance magic value.
*/
function safeBatchTransferFrom(address from, address to, uint256[] calldata ids,
uint256[] calldata amounts, bytes calldata data) external;
}

// File:
openzeppelin-contracts-master/contracts/token/ERC1155/IERC1155MetadataURI.sol

pragma solidity ^0.6.2;

/**
* @dev Interface of the optional ERC1155MetadataExtension interface, as defined
* in the https://eips.ethereum.org/EIPS/eip-1155#metadata-extensions[EIP].
*
* _Available since v3.1._
*/
interface IERC1155MetadataURI is IERC1155 {
/**
* @dev Returns the URI for token type `id`.
*
* If the `\{id\}` substring is present in the URI, it must be replaced by
* clients with the actual token type ID.
*/
function uri(uint256 id) external view returns (string memory);
}

// File:
openzeppelin-contracts-master/contracts/token/ERC1155/IERC1155Receiver.sol

pragma solidity ^0.6.2;
```

```
/**
* _Available since v3.1._
*/
interface IERC1155Receiver is IERC165 {

/**
@dev Handles the receipt of a single ERC1155 token type.This function is
called at the end of a `safeTransferFrom` after the balance has been updated.
To acc Block World    the transfer, this must return
`bytes4(keccak256("onERC1155Received(address,address,uint256,uint256,bytes)"))`
(i.e.0xf23a6e61, or its own function selector).
@param operator The address which initiated the transfer (i.e.msg.sender)
@param from The address which previously owned the token
@param id The ID of the token being transferred
@param value The amount of tokens being transferred
@param data Additional data with no specified format
@return
`bytes4(keccak256("onERC1155Received(address,address,uint256,uint256,bytes)"))`
if transfer is allowed
*/
function onERC1155Received(
address operator,
address from,
uint256 id,
uint256 value,
bytes calldata data
)
external
returns(bytes4);

/**
@dev Handles the receipt of a multiple ERC1155 token types.This function
is called at the end of a `safeBatchTransferFrom` after the balances have
been updated.To acc Block World    the transfer(s), this must return
`bytes4(keccak256("onERC1155BatchReceived(address,address,uint256[],uint256[],b
ytes)"))`
(i.e.0xbc197c81, or its own function selector).
@param operator The address which initiated the batch transfer (i.e.msg.sender)
@param from The address which previously owned the token
@param ids An array containing ids of each token being transferred (order and
length must match values array)
@param values An array containing amounts of each token being transferred (order
and length must match ids array)
@param data Additional data with no specified format
@return
`bytes4(keccak256("onERC1155BatchReceived(address,address,uint256[],uint256[],b
ytes)"))` if transfer is allowed
*/
function onERC1155BatchReceived(
address operator,
address from,
uint256[] calldata ids,
uint256[] calldata values,
bytes calldata data
)
external
returns(bytes4);
}

// File: openzeppelin-contracts-master/contracts/GSN/Context.sol

pragma solidity ^0.6.2;

/*
* @dev Provides information about the current execution context, including the
* sender of the transaction and its data.While these are generally available
* via msg.sender and msg.data, they should not be accessed in such a direct
* manner, since when dealing with GSN meta-transactions the account sending and
* paying for execution may not be the actual sender (as far as an application
* is concerned).
*
* This contract is only required for intermediate, library-like contracts.
*/
abstract contract Context {
function _msgSender() internal view virtual returns (address payable) {
return msg.sender;
}

function _msgData() internal view virtual returns (bytes memory) {
this; // silence state mutability warning without generating bytecode - see
https://github.com/ethereum/solidity/issues/2691
return msg.data;
}
}

// File: openzeppelin-contracts-master/contracts/introspection/ERC165.sol
```

```
pragma solidity ^0.6.2;

/**
* @dev Implementation of the {IERC165} interface.
*
* Contracts may inherit from this and call {_registerInterface} to declare
* their support of an interface.
*/
contract ERC165 is IERC165 {
/*
* bytes4(keccak256('supportsInterface(bytes4)')) == 0x01ffc9a7
*/
bytes4 private constant _INTERFACE_ID_ERC165 = 0x01ffc9a7;

/**
* @dev Mapping of interface ids to whether or not it's supported.
*/
mapping(bytes4 => bool) private _supportedInterfaces;

constructor () internal {
// Derived contracts need only register support for their own interfaces,
// we register support for ERC165 itself here
_registerInterface(_INTERFACE_ID_ERC165);
}

/**
* @dev See {IERC165-supportsInterface}.
*
* Time complexity O(1), guaranteed to always use less than 30 000 gas.
*/
function supportsInterface(bytes4 interfaceId) public view override returns (bool) {
return _supportedInterfaces[interfaceId];
}

/**
* @dev Registers the contract as an implementer of the interface defined by
* `interfaceId`.Support of the actual ERC165 interface is automatic and
* registering its interface id is not required.
*
* See {IERC165-supportsInterface}.
*
* Requirements:
*
* - `interfaceId` cannot be the ERC165 invalid interface (`0xffffffff`).
```

```
*/
function _registerInterface(bytes4 interfaceId) internal virtual {
require(interfaceId != 0xffffffff, "ERC165: invalid interface id");
_supportedInterfaces[interfaceId] = true;
}
}

// File: openzeppelin-contracts-master/contracts/math/SafeMath.sol

pragma solidity ^0.6.2;

/**
* @dev Wrappers over Solidity's arithmetic operations with added overflow
* checks.
*
* Arithmetic operations in Solidity wrap on overflow.This can easily result
* in bugs, because programmers usually assume that an overflow raises an
* error, which is the standard behavior in high level programming languages.
* `SafeMath` restores this intuition by reverting the transaction when an
* operation overflows.
*
* Using this library instead of the unchecked operations eliminates an entire
* class of bugs, so it's recommended to use it always.
*/
library SafeMath {
/**
* @dev Returns the addition of two unsigned integers, reverting on
* overflow.
*
* Counterpart to Solidity's `+` operator.
*
* Requirements:
*
* - Addition cannot overflow.
*/
function add(uint256 a, uint256 b) internal pure returns (uint256) {
uint256 c = a + b;
require(c >= a, "SafeMath: addition overflow");

return c;
}

/**
* @dev Returns the subtraction of two unsigned integers, reverting on
* overflow (when the result is negative).
*
* Counterpart to Solidity's `-` operator.
*
* Requirements:
*
* - Subtraction cannot overflow.
*/
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
return sub(a, b, "SafeMath: subtraction overflow");
}

/**
* @dev Returns the subtraction of two unsigned integers, reverting with custom
message on
* overflow (when the result is negative).
*
* Counterpart to Solidity's `-` operator.
*
* Requirements:
*
* - Subtraction cannot overflow.
*/
function sub(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
require(b <= a, errorMessage);
uint256 c = a - b;

return c;
}

/**
* @dev Returns the multiplication of two unsigned integers, reverting on
* overflow.
*
* Counterpart to Solidity's `*` operator.
*
* Requirements:
*
* - Multiplication cannot overflow.
*/
function mul(uint256 a, uint256 b) internal pure returns (uint256) {
// Gas optimization: this is cheaper than requiring 'a' not being zero, but the
// benefit is lost if 'b' is also tested.
// See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
if (a == 0) {
return 0;
}

uint256 c = a * b;
require(c / a == b, "SafeMath: multiplication overflow");

return c;
}

/**
* @dev Returns the integer division of two unsigned integers.Reverts on
* division by zero.The result is rounded towards zero.
*
* Counterpart to Solidity's `/` operator.Note: this function uses a
* `revert` opcode (which leaves remaining gas untouched) while Solidity
* uses an invalid opcode to revert (consuming all remaining gas).
*
* Requirements:
*
* - The divisor cannot be zero.
*/
function div(uint256 a, uint256 b) internal pure returns (uint256) {
return div(a, b, "SafeMath: division by zero");
}

/**
* @dev Returns the integer division of two unsigned integers.Reverts with custom
message on
* division by zero.The result is rounded towards zero.
*
* Counterpart to Solidity's `/` operator.Note: this function uses a
* `revert` opcode (which leaves remaining gas untouched) while Solidity
* uses an invalid opcode to revert (consuming all remaining gas).
*
* Requirements:
*
* - The divisor cannot be zero.
*/
function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
require(b > 0, errorMessage);
uint256 c = a / b;
```

```
// assert(a == b * c + a % b); // There is no case in which this doesn't hold

return c;
}

/**
* @dev Returns the remainder of dividing two unsigned integers.(unsigned integer modulo),
* Reverts when dividing by zero.
*
* Counterpart to Solidity's `%` operator.This function uses a `revert`
* opcode (which leaves remaining gas untouched) while Solidity uses an
* invalid opcode to revert (consuming all remaining gas).
*
* Requirements:
*
* - The divisor cannot be zero.
*/
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
return mod(a, b, "SafeMath: modulo by zero");
}

/**
* @dev Returns the remainder of dividing two unsigned integers.(unsigned integer modulo),
* Reverts with custom message when dividing by zero.
*
* Counterpart to Solidity's `%` operator.This function uses a `revert`
* opcode (which leaves remaining gas untouched) while Solidity uses an
* invalid opcode to revert (consuming all remaining gas).
*
* Requirements:
*
* - The divisor cannot be zero.
*/
function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
require(b != 0, errorMessage);
return a % b;
}
}

// File: openzeppelin-contracts-master/contracts/utils/Address.sol
```

```
pragma solidity ^0.6.2;

/**
* @dev Collection of functions related to the address type
*/
library Address {
/**
* @dev Returns true if `account` is a contract.
*
* [IMPORTANT]
* ====
* It is unsafe to assume that an address for which this function returns
* false is an externally-owned account (EOA) and not a contract.
*
* Among others, `isContract` will return false for the following
* types of addresses:
*
*   - an externally-owned account
*   - a contract in construction
*   - an address where a contract will be created
*   - an address where a contract lived, but was destroyed
* ====
*/
function isContract(address account) internal view returns (bool) {
// This method relies on extcodesize, which returns 0 for contracts in
// construction, since the code is only stored at the end of the
// constructor execution.

uint256 size;
// solhint-disable-next-line no-inline-assembly
assembly { size := extcodesize(account) }
return size > 0;
}

/**
* @dev Replacement for Solidity's `transfer`: sends `amount` wei to
* `recipient`, forwarding all available gas and reverting on errors.
*
* https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
* of certain opcodes, possibly making contracts go over the 2300 gas limit
* imposed by `transfer`, making them unable to receive funds via
* `transfer`.{sendValue} removes this limitation.
*
*
```

```
https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Le
arn more].
*
* IMPORTANT: because control is transferred to `recipient`, care must be
* taken to not create reentrancy vulnerabilities.Consider using
* {ReentrancyGuard} or the
*
https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-che
cks-effects-interactions-pattern[checks-effects-interactions pattern].
*/
function sendValue(address payable recipient, uint256 amount) internal {
require(address(this).balance >= amount, "Address: insufficient balance");

// solhint-disable-next-line avoid-low-level-calls, avoid-call-value
(bool success, ) = recipient.call{ value: amount }("");
require(success, "Address: unable to send value, recipient may have reverted");
}

/**
* @dev Performs a Solidity function call using a low level `call`.A
* plain`call` is an unsafe replacement for a function call: use this
* function instead.
*
* If `target` reverts with a revert reason, it is bubbled up by this
* function (like regular Solidity function calls).
*
* Returns the raw returned data.To convert to the expected return value,
*                                                                           use
https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=a
bi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
*
* Requirements:
*
* - `target` must be a contract.
* - calling `target` with `data` must not revert.
*
* _Available since v3.1._
*/
function functionCall(address target, bytes memory data) internal returns (bytes
memory) {
return functionCall(target, data, "Address: low-level call failed");
}

/**
```

```
* @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
* `errorMessage` as a fallback revert reason when `target` reverts.
*
* _Available since v3.1._
*/
function functionCall(address target, bytes memory data, string memory
errorMessage) internal returns (bytes memory) {
return functionCallWithValue(target, data, 0, errorMessage);
}

/**
* @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
* but also transferring `value` wei to `target`.
*
* Requirements:
*
* - the calling contract must have an ETH balance of at least `value`.
* - the called Solidity function must be `payable`.
*
* _Available since v3.1._
*/
function functionCallWithValue(address target, bytes memory data, uint256 value)
internal returns (bytes memory) {
return functionCallWithValue(target, data, value, "Address: low-level call with value
failed");
}

/**
* @dev Same as
{xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue
`], but
* with `errorMessage` as a fallback revert reason when `target` reverts.
*
* _Available since v3.1._
*/
function functionCallWithValue(address target, bytes memory data, uint256 value,
string memory errorMessage) internal returns (bytes memory) {
require(address(this).balance >= value, "Address: insufficient balance for call");
require(isContract(target), "Address: call to non-contract");

// solhint-disable-next-line avoid-low-level-calls
(bool success, bytes memory returndata) = target.call{ value: value }(data);
return _verifyCallResult(success, returndata, errorMessage);
}
```

```
/**
* @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
* but performing a static call.
*
* _Available since v3.3._
*/
function functionStaticCall(address target, bytes memory data) internal view returns
(bytes memory) {
return functionStaticCall(target, data, "Address: low-level static call failed");
}

/**
* @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
* but performing a static call.
*
* _Available since v3.3._
*/
function functionStaticCall(address target, bytes memory data, string memory
errorMessage) internal view returns (bytes memory) {
require(isContract(target), "Address: static call to non-contract");

// solhint-disable-next-line avoid-low-level-calls
(bool success, bytes memory returndata) = target.staticcall(data);
return _verifyCallResult(success, returndata, errorMessage);
}

/**
* @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
* but performing a delegate call.
*
* _Available since v3.3._
*/
function functionDelegateCall(address target, bytes memory data) internal returns
(bytes memory) {
return functionDelegateCall(target, data, "Address: low-level delegate call failed");
}

/**
* @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
* but performing a delegate call.
*
* _Available since v3.3._
*/
```

```
function functionDelegateCall(address target, bytes memory data, string memory
errorMessage) internal returns (bytes memory) {
require(isContract(target), "Address: delegate call to non-contract");

// solhint-disable-next-line avoid-low-level-calls
(bool success, bytes memory returndata) = target.delegatecall(data);
return _verifyCallResult(success, returndata, errorMessage);
}

function _verifyCallResult(bool success, bytes memory returndata, string memory
errorMessage) private pure returns(bytes memory) {
if (success) {
return returndata;
} else {
// Look for revert reason and bubble it up if present
if (returndata.length > 0) {
// The easiest way to bubble the revert reason is using memory via assembly

// solhint-disable-next-line no-inline-assembly
assembly {
let returndata_size := mload(returndata)
revert(add(32, returndata), returndata_size)
}
} else {
revert(errorMessage);
}
}
}
}

// @dev Implementation for different URIs for every token
contract TokenURI {
// mapping for token URIs
mapping(uint256 => string) private _tokenURIs;

function _tokenURI(uint256 tokenId) internal view returns (string memory) {
return _tokenURIs[tokenId];
}

function _setTokenURI(uint256 tokenId, string memory tokenUri) virtual internal {
_tokenURIs[tokenId] = tokenUri;
}
}
```

```
// File: openzeppelin-contracts-master/contracts/token/ERC1155/ERC1155.sol
pragma solidity ^0.6.2;

/**
*
* @dev Implementation of the basic standard multi-token.
* See https://eips.ethereum.org/EIPS/eip-1155
* Originally based on code by Enjin: https://github.com/enjin/erc-1155
*
* _Available since v3.1._
*/
contract ERC1155 is Context, ERC165, IERC1155, IERC1155MetadataURI, TokenURI {
using SafeMath for uint256;
using Address for address;

// Mapping from token ID to account balances
mapping (uint256 => mapping(address => uint256)) private _balances;

// Mapping from account to operator approvals
mapping (address => mapping(address => bool)) private _operatorApprovals;

// Used as the URI for all token types by relying on ID substitution,
e.g.https://token-cdn-domain/{id}.json
string private _uri;

/*
*       bytes4(keccak256('balanceOf(address,uint256)')) == 0x00fdd58e
*       bytes4(keccak256('balanceOfBatch(address[],uint256[])')) == 0x4e1273f4
*       bytes4(keccak256('setApprovalForAll(address,bool)')) == 0xa22cb465
*       bytes4(keccak256('isApprovedForAll(address,address)')) == 0xe985e9c5
*        bytes4(keccak256('safeTransferFrom(address,address,uint256,uint256,bytes)'))
== 0xf242432a
*       bytes4(keccak256('safeBatchTransferFrom(address,address,uint256[],uint256[],
bytes)')) == 0x2eb2c2d6
*
*       => 0x00fdd58e ^ 0x4e1273f4 ^ 0xa22cb465 ^
*           0xe985e9c5 ^ 0xf242432a ^ 0x2eb2c2d6 == 0xd9b67a26
*/
bytes4 private constant _INTERFACE_ID_ERC1155 = 0xd9b67a26;

/*
*       bytes4(keccak256('uri(uint256)')) == 0x0e89341c
*/
bytes4 private constant _INTERFACE_ID_ERC1155_METADATA_URI = 0x0e89341c;
```

```
/**
* @dev See {_setURI}.
*/
constructor (string memory uri) public {
_setURI(uri);

// register the supported interfaces to conform to ERC1155 via ERC165
_registerInterface(_INTERFACE_ID_ERC1155);

// register the supported interfaces to conform to ERC1155MetadataURI via ERC165
_registerInterface(_INTERFACE_ID_ERC1155_METADATA_URI);
}

/**
* @dev See {IERC1155MetadataURI-uri}.
*
* This implementation returns the same URI for *all* token types.It relies
* on the token type ID substitution mechanism
* https://eips.ethereum.org/EIPS/eip-1155#metadata[defined in the EIP].
*
* Clients calling this function must replace the `\{id\}` substring with the
* actual token type ID.
*/
function uri(uint256 id) external view virtual override returns (string memory) {
return _tokenURI(id);
}

/**
* @dev See {IERC1155-balanceOf}.
*
* Requirements:
*
* - `account` cannot be the zero address.
*/
function balanceOf(address account, uint256 id) public view override returns
(uint256) {
require(account != address(0), "ERC1155: balance query for the zero address");
return _balances[id][account];
}

/**
* @dev See {IERC1155-balanceOfBatch}.
*
```

```
* Requirements:
*
* - `accounts` and `ids` must have the same length.
*/
function balanceOfBatch(
address[] memory accounts,
uint256[] memory ids
)
public
view
override
returns (uint256[] memory)
{
require(accounts.length == ids.length, "ERC1155: accounts and ids length
mismatch");

uint256[] memory batchBalances = new uint256[](accounts.length);

for (uint256 i = 0; i < accounts.length; ++i) {
require(accounts[i] != address(0), "ERC1155: batch balance query for the zero
address");
batchBalances[i] = _balances[ids[i]][accounts[i]];
}

return batchBalances;
}

/**
* @dev See {IERC1155-setApprovalForAll}.
*/
function setApprovalForAll(address operator, bool approved) public virtual override {
require(_msgSender() != operator, "ERC1155: setting approval status for self");

_operatorApprovals[_msgSender()][operator] = approved;
emit ApprovalForAll(_msgSender(), operator, approved);
}

/**
* @dev See {IERC1155-isApprovedForAll}.
*/
function isApprovedForAll(address account, address operator) public view override
returns (bool) {
return _operatorApprovals[account][operator];
}
```

```
/**
* @dev See {IERC1155-safeTransferFrom}.
*/
function safeTransferFrom(
address from,
address to,
uint256 id,
uint256 amount,
bytes memory data
)
public
virtual
override
{
require(to != address(0), "ERC1155: transfer to the zero address");
require(
from == _msgSender() || isApprovedForAll(from, _msgSender()),
"ERC1155: caller is not owner nor approved"
);

address operator = _msgSender();

_beforeTokenTransfer(operator, from, to, _asSingletonArray(id),
_asSingletonArray(amount), data);

_balances[id][from] = _balances[id][from].sub(amount, "ERC1155: insufficient
balance for transfer");
_balances[id][to] = _balances[id][to].add(amount);

emit TransferSingle(operator, from, to, id, amount);

_doSafeTransferAcc Block World anceCheck(operator, from, to, id, amount, data);
}

/**
* @dev See {IERC1155-safeBatchTransferFrom}.
*/
function safeBatchTransferFrom(
address from,
address to,
uint256[] memory ids,
uint256[] memory amounts,
bytes memory data
)
public
virtual
override
{
require(ids.length == amounts.length, "ERC1155: ids and amounts length mismatch");
require(to != address(0), "ERC1155: transfer to the zero address");
require(
from == _msgSender() || isApprovedForAll(from, _msgSender()),
"ERC1155: transfer caller is not owner nor approved"
);

address operator = _msgSender();

_beforeTokenTransfer(operator, from, to, ids, amounts, data);

for (uint256 i = 0; i < ids.length; ++i) {
uint256 id = ids[i];
uint256 amount = amounts[i];

_balances[id][from] = _balances[id][from].sub(
amount,
"ERC1155: insufficient balance for transfer"
);
_balances[id][to] = _balances[id][to].add(amount);
}

emit TransferBatch(operator, from, to, ids, amounts);

_doSafeBatchTransferAcc Block World anceCheck(operator, from, to, ids, amounts, data);
}

/**
* @dev Sets a new URI for all token types, by relying on the token type ID
* substitution mechanism
* https://eips.ethereum.org/EIPS/eip-1155#metadata[defined in the EIP].
*
* By this mechanism, any occurrence of the `\{id\}` substring in either the
* URI or any of the amounts in the JSON file at said URI will be replaced by
* clients with the token type ID.
*
* For example, the `https://token-cdn-domain/\{id\}.json` URI would be
* interpreted by clients as
*
`https://token-cdn-domain/000000000000000000000000000000000000000000000000000000000004cce0.json`
* for token type ID 0x4cce0.
*
* See {uri}.
*
* Because these URIs cannot be meaningfully represented by the {URI} event,
* this function emits no events.
*/
function _setURI(string memory newuri) internal virtual {
_uri = newuri;
}

/**
* @dev Creates `amount` tokens of token type `id`, and assigns them to `account`.
*
* Emits a {TransferSingle} event.
*
* Requirements:
*
* - `account` cannot be the zero address.
* - If `to` refers to a smart contract, it must implement {IERC1155Receiver-onERC1155Received} and return the
* acc Block World ance magic value.
*/
function _mint(address account, uint256 tokenId, uint256 amount, string memory tokenUri, bytes memory data) internal virtual {
require(account != address(0), "ERC1155: mint to the zero address");

address operator = _msgSender();

_beforeTokenTransfer(operator, address(0), account, _asSingletonArray(tokenId), _asSingletonArray(amount), data);

_balances[tokenId][account] = _balances[tokenId][account].add(amount);
_setTokenURI(tokenId, tokenUri);

emit TransferSingle(operator, address(0), account, tokenId, amount);

_doSafeTransferAcc Block World anceCheck(operator, address(0), account, tokenId, amount, data);
}
```

```
/**
* @dev Internal function to set the token URI for a given token.
* Reverts if the token ID does not exist.
* @param tokenId uint256 ID of the token to set its URI
* @param tokenUri string URI to assign
*/
function _setURI( uint256 tokenId,   string memory tokenUri) internal virtual {
_setTokenURI(tokenId, tokenUri);
}

function _setTokenURI(uint256 tokenId, string memory tokenUri) internal override {
super._setTokenURI(tokenId, tokenUri);
}

/**
* @dev xref:ROOT:erc1155.adoc#batch-operations[Batched] version of {_mint}.
*
* Requirements:
*
* - `ids` and `amounts` must have the same length.
* - If `to` refers to a smart contract, it must implement {IERC1155Receiver-onERC1155BatchReceived} and return the
* acc Block World ance magic value.
*/
function _mintBatch(address to, uint256[] memory ids, uint256[] memory amounts, bytes memory data) internal virtual {
require(to != address(0), "ERC1155: mint to the zero address");
require(ids.length == amounts.length, "ERC1155: ids and amounts length mismatch");

address operator = _msgSender();

_beforeTokenTransfer(operator, address(0), to, ids, amounts, data);

for (uint i = 0; i < ids.length; i++) {
_balances[ids[i]][to] = amounts[i].add(_balances[ids[i]][to]);
}

emit TransferBatch(operator, address(0), to, ids, amounts);

_doSafeBatchTransferAcc Block World anceCheck(operator, address(0), to, ids, amounts, data);
}
```

```
/**
* @dev Destroys `amount` tokens of token type `id` from `account`
*
* Requirements:
*
* - `account` cannot be the zero address.
* - `account` must have at least `amount` tokens of token type `id`.
*/
function _burn(address account, uint256 id, uint256 amount) internal virtual {
require(account != address(0), "ERC1155: burn from the zero address");

address operator = _msgSender();

_beforeTokenTransfer(operator, account, address(0), _asSingletonArray(id),
_asSingletonArray(amount), "");

_balances[id][account] = _balances[id][account].sub(
amount,
"ERC1155: burn amount exceeds balance"
);

emit TransferSingle(operator, account, address(0), id, amount);
}

/**
* @dev xref:ROOT:erc1155.adoc#batch-operations[Batched] version of {_burn}.
*
* Requirements:
*
* - `ids` and `amounts` must have the same length.
*/
function _burnBatch(address account, uint256[] memory ids, uint256[] memory
amounts) internal virtual {
require(account != address(0), "ERC1155: burn from the zero address");
require(ids.length == amounts.length, "ERC1155: ids and amounts length
mismatch");

address operator = _msgSender();

_beforeTokenTransfer(operator, account, address(0), ids, amounts, "");

for (uint i = 0; i < ids.length; i++) {
_balances[ids[i]][account] = _balances[ids[i]][account].sub(
amounts[i],
"ERC1155: burn amount exceeds balance"
);
}

emit TransferBatch(operator, account, address(0), ids, amounts);
}

/**
* @dev Hook that is called before any token transfer.This includes minting
* and burning, as well as batched variants.
*
* The same hook is called on both single and batched variants.For single
* transfers, the length of the `id` and `amount` arrays will be 1.
*
* Calling conditions (for each `id` and `amount` pair):
*
* - When `from` and `to` are both non-zero, `amount` of ``from``'s tokens
* of token type `id` will be   transferred to `to`.
* - When `from` is zero, `amount` tokens of token type `id` will be minted
* for `to`.
* - when `to` is zero, `amount` of ``from``'s tokens of token type `id`
* will be burned.
* - `from` and `to` are never both zero.
* - `ids` and `amounts` have the same, non-zero length.
*
* To learn more about hooks, head to
xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
*/
function _beforeTokenTransfer(
address operator,
address from,
address to,
uint256[] memory ids,
uint256[] memory amounts,
bytes memory data
)
internal virtual
{ }

function _doSafeTransferAcc Block World anceCheck(
address operator,
address from,
address to,
uint256 id,
uint256 amount,
bytes memory data
)
private
{
if (to.isContract()) {
try IERC1155Receiver(to).onERC1155Received(operator, from, id, amount, data)
returns (bytes4 response) {
if (response != IERC1155Receiver(to).onERC1155Received.selector) {
revert("ERC1155: ERC1155Receiver rejected tokens");
}
} catch Error(string memory reason) {
revert(reason);
} catch {
revert("ERC1155: transfer to non ERC1155Receiver implementer");
}
}
}

function _doSafeBatchTransferAcc Block World anceCheck(
address operator,
address from,
address to,
uint256[] memory ids,
uint256[] memory amounts,
bytes memory data
)
private
{
if (to.isContract()) {
try IERC1155Receiver(to).onERC1155BatchReceived(operator, from, ids, amounts,
data) returns (bytes4 response) {
if (response != IERC1155Receiver(to).onERC1155BatchReceived.selector) {
revert("ERC1155: ERC1155Receiver rejected tokens");
}
} catch Error(string memory reason) {
revert(reason);
} catch {
revert("ERC1155: transfer to non ERC1155Receiver implementer");
}
}
}
```

```
function _asSingletonArray(uint256 element) private pure returns (uint256[] memory)
{
uint256[] memory array = new uint256[](1);
array[0] = element;

return array;
}
}

pragma solidity ^0.6.2;

contract NFTTokens is ERC1155 {
address public governance;
uint256 public nftCount;

modifier onlyGovernance() {
require(msg.sender == governance, "only governance can call this");

_;
}

constructor(address governance_) public ERC1155("MateXgame") {
governance = governance_;
nftCount = 0;
}

function addNewcard(uint256 initialSupply,string calldata _tokenUri) external
onlyGovernance {
nftCount++;
uint256 nftTokenClassId = nftCount;
_mint(msg.sender, nftTokenClassId, initialSupply, _tokenUri, "");
}

function getMetaverseCount() public view   returns(uint256) {
return nftCount;
}
function setURI( uint256 tokenId,string calldata tokenUri) external onlyGovernance {
_setURI(tokenId, tokenUri);
}
}
```

0x6 Reference

0x61.Fabian Vogelsteller and Vitalik Buterin.Nov.2015.

https://eips.ethereum.org/EIPS/eip-20

0x 62.A Hands-On Tutorial for Zero-Knowledge Proofs.

https://en.wikipedia.org/wiki/Zero-knowledge_proof

0x63.ERC-20 (Ethereum Consultation 20) was proposed by Fabian Vogelsteller in November 2015.

https://ethereum.org/zh/developers/docs/standards/tokens/erc-20/

0x64. "How to Create an NFT – Simply Explained".Eduwab.May 19, 2022. Retrieved June 3, 2022.

0x65.The Rise of Non-Fungible Tokens : Exploring the Future of Digital Collectibles" by Chris Burns.

0x66.Non-Fungible Tokens: A Survey" by C.Szegedi et al.