Contract 0x6a6d5Dc29ad8ff23209186775873e123b31c26E9

Buy Play Gaming

Source Code

Overview

POL BALANCE

0 POL

POL VALUE

$0.00

TOKEN HOLDINGS

$0.00 (1 Tokens)

More Info

PRIVATE NAME TAGS

+ Add

CONTRACT CREATOR

0xf9909c6C...6ae334Ea3 at txn 0xd1305c3595...

Multichain Info

$0 (Multichain Portfolio)

No addresses found

Transactions Token Transfers (ERC-20) Contract Events Assets

Code Read Contract Write Contract

Search Source Code

Contract Source Code Verified (Exact Match)

Contract Name:	**LHILecceNFT**
Compiler Version	**v0.8.20+commit.a1b79de6**
Optimization Enabled:	**No** with **200** runs
Other Settings:	**paris** EvmVersion

Contract Source Code (Solidity Standard Json-Input format)

IDE More Options

File 1 of 21 : LHILecceNFT.sol

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

import "@openzeppelin/contracts/token/ERC1155/extensions/ERC1155URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract LHILecceNFT is ERC1155URIStorage, Ownable {
    string public name = "LHI Lecce NFT";
    string public symbol = "LHILE";

    mapping(uint256 => uint256) public maxSupply;
    mapping(uint256 => uint256) public totalMinted;
    mapping(uint256 => uint256) public pricesInWei; // Prezzi in wei
    mapping(uint256 => bool) public isValidTokenId; // Per controllare i tokenId validi
    mapping(uint256 => string) private encryptedURIs; // Mappa tokenId agli URI crittog
    mapping(uint256 => string) private tokenCIDs; // Mappa tokenId ai CIDs

    address public withdrawWallet;

    struct BurnRequest {
        address requester;
        uint256 tokenId;
        uint256 quantity;
        bool approved;
    }
```

File 2 of 21 : Ownable.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.0.0) (access/Ownable.sol)

pragma solidity ^0.8.20;

import {Context} from "../utils/Context.sol";

/**
 * @dev Contract module which provides a basic access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * The initial owner is set to the address provided by the deployer. This can
 * later be changed with {transferOwnership}.
 *
```

```
 * This module is used through inheritance. It will make available the modifier
 * `onlyOwner`, which can be applied to your functions to restrict their use to
 * the owner.
 */
abstract contract Ownable is Context {
    address private _owner;

    /**
     * @dev The caller account is not authorized to perform an operation.
```

File 3 of 21 : draft-IERC6093.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (interfaces/draft-IERC6093.sol)
pragma solidity ^0.8.20;

/**
 * @dev Standard ERC-20 Errors
 * Interface of the https://eips.ethereum.org/EIPS/eip-6093[ERC-6093] custom errors for
 */
interface IERC20Errors {
    /**
     * @dev Indicates an error related to the current `balance` of a `sender`. Used in
     * @param sender Address whose tokens are being transferred.
     * @param balance Current balance for the interacting account.
     * @param needed Minimum amount required to perform a transfer.
     */
    error ERC20InsufficientBalance(address sender, uint256 balance, uint256 needed);

    /**
     * @dev Indicates a failure with the token `sender`. Used in transfers.
     * @param sender Address whose tokens are being transferred.
     */
    error ERC20InvalidSender(address sender);

    /**
     * @dev Indicates a failure with the token `receiver`. Used in transfers.
```

File 4 of 21 : ERC1155.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (token/ERC1155/ERC1155.sol)

pragma solidity ^0.8.20;

import {IERC1155} from "./IERC1155.sol";
import {IERC1155MetadataURI} from "./extensions/IERC1155MetadataURI.sol";
import {ERC1155Utils} from "./utils/ERC1155Utils.sol";
import {Context} from "../../utils/Context.sol";
import {IERC165, ERC165} from "../../utils/introspection/ERC165.sol";
import {Arrays} from "../../utils/Arrays.sol";
import {IERC1155Errors} from "../../interfaces/draft-IERC6093.sol";

/**
 * @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
 */
abstract contract ERC1155 is Context, ERC165, IERC1155, IERC1155MetadataURI, IERC1155Er
    using Arrays for uint256[];
    using Arrays for address[];

    mapping(uint256 id => mapping(address account => uint256)) private _balances;

    mapping(address account => mapping(address operator => bool)) private  operatorAppr
```

File 5 of 21 : ERC1155URIStorage.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (token/ERC1155/extensions/ERC1155URIStor

```

```solidity
pragma solidity ^0.8.20;

import {Strings} from "../../../utils/Strings.sol";
import {ERC1155} from "../ERC1155.sol";

/**
 * @dev ERC-1155 token with storage based token URI management.
 * Inspired by the {ERC721URIStorage} extension
 */
abstract contract ERC1155URIStorage is ERC1155 {
    using Strings for uint256;

    // Optional base URI
    string private _baseURI = "";

    // Optional mapping for token URIs
    mapping(uint256 tokenId => string) private _tokenURIs;

    /**
     * @dev See {IERC1155MetadataURI-uri}.
     *
```

File 6 of 21 : IERC1155MetadataURI.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (token/ERC1155/extensions/IERC1155MetadataU

pragma solidity ^0.8.20;

import {IERC1155} from "../IERC1155.sol";

/**
 * @dev Interface of the optional ERC1155MetadataExtension interface, as defined
 * in the https://eips.ethereum.org/EIPS/eip-1155#metadata-extensions[ERC].
 */
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 7 of 21 : IERC1155.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.3.0) (token/ERC1155/IERC1155.sol)

pragma solidity ^0.8.20;

import {IERC165} from "../../utils/introspection/IERC165.sol";

/**
 * @dev Required interface of an ERC-1155 compliant contract, as defined in the
 * https://eips.ethereum.org/EIPS/eip-1155[ERC].
 */
interface IERC1155 is IERC165 {
    /**
     * @dev Emitted when `value` amount of tokens of type `id` are transferred from `fr
     */
    event TransferSingle(address indexed operator, address indexed from, address indexe

    /**
     * @dev Equivalent to multiple {TransferSingle} events, where `operator`, `from` an
     * transfers.
     */
    event TransferBatch(
        address indexed operator,
        address indexed from,
        address indexed to,
```

File 8 of 21 : IERC1155Receiver.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (token/ERC1155/IERC1155Receiver.sol)

pragma solidity ^0.8.20;

import {IERC165} from "../../utils/introspection/IERC165.sol";

/**
 * @dev Interface that must be implemented by smart contracts in order to receive
 * ERC-1155 token transfers.
 */
interface IERC1155Receiver is IERC165 {
    /**
     * @dev Handles the receipt of a single ERC-1155 token type. This function is
     * called at the end of a `safeTransferFrom` after the balance has been updated.
     *
     * NOTE: To accept 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
```

File 9 of 21 : ERC1155Utils.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.3.0) (token/ERC1155/utils/ERC1155Utils.sol)

pragma solidity ^0.8.20;

import {IERC1155Receiver} from "../IERC1155Receiver.sol";
import {IERC1155Errors} from "../../../interfaces/draft-IERC6093.sol";

/**
 * @dev Library that provide common ERC-1155 utility functions.
 *
 * See https://eips.ethereum.org/EIPS/eip-1155[ERC-1155].
 *
 * _Available since v5.1._
 */
library ERC1155Utils {
    /**
     * @dev Performs an acceptance check for the provided `operator` by calling {IERC115
     * on the `to` address. The `operator` is generally the address that initiated the t
     *
     * The acceptance call is not executed and treated as a no-op if the target address
     * Otherwise, the recipient must implement {IERC1155Receiver-onERC1155Received} and
     * the transfer.
     */
    function checkOnERC1155Received(
```

File 10 of 21 : Arrays.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.3.0) (utils/Arrays.sol)
// This file was procedurally generated from scripts/generate/templates/Arrays.js.

pragma solidity ^0.8.20;

import {Comparators} from "./Comparators.sol";
import {SlotDerivation} from "./SlotDerivation.sol";
import {StorageSlot} from "./StorageSlot.sol";
import {Math} from "./math/Math.sol";

/**
 * @dev Collection of functions related to array types.
```

```
 */
library Arrays {
    using SlotDerivation for bytes32;
    using StorageSlot for bytes32;

    /**
     * @dev Sort an array of uint256 (in memory) following the provided comparator func
     *
     * This function does the sorting "in place", meaning that it overrides the input.
     * convenience, but that returned value can be discarded safely if the caller has a
     *
```

File 11 of 21 : Comparators.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (utils/Comparators.sol)

pragma solidity ^0.8.20;

/**
 * @dev Provides a set of functions to compare values.
 *
 * _Available since v5.1._
 */
library Comparators {
    function lt(uint256 a, uint256 b) internal pure returns (bool) {
        return a < b;
    }

    function gt(uint256 a, uint256 b) internal pure returns (bool) {
        return a > b;
    }
}
```

File 12 of 21 : Context.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.0.1) (utils/Context.sol)

pragma solidity ^0.8.20;

/**
 * @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 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) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }

    function  contextSuffixLength() internal view virtual returns (uint256) {
```

File 13 of 21 : ERC165.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (utils/introspection/ERC165.sol)

pragma solidity ^0.8.20;

import {IERC165} from "./IERC165.sol";

```

```solidity
/**
 * @dev Implementation of the {IERC165} interface.
 *
 * Contracts that want to implement ERC-165 should inherit from this contract and overri
 * for the additional interface id that will be supported. For example:
 *
 * ```solidity
 * function supportsInterface(bytes4 interfaceId) public view virtual override returns (
 *     return interfaceId == type(MyInterface).interfaceId || super.supportsInterface(in
 * }
 * ```
 */
abstract contract ERC165 is IERC165 {
    /**
     * @dev See {IERC165-supportsInterface}.
     */
    function supportsInterface(bytes4 interfaceId) public view virtual returns (bool) {
        return interfaceId == type(IERC165).interfaceId;
```

File 14 of 21 : IERC165.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (utils/introspection/IERC165.sol)

pragma solidity ^0.8.20;

/**
 * @dev Interface of the ERC-165 standard, as defined in the
 * https://eips.ethereum.org/EIPS/eip-165[ERC].
 *
 * 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[ERC 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 15 of 21 : Math.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.3.0) (utils/math/Math.sol)

pragma solidity ^0.8.20;

import {Panic} from "../Panic.sol";
import {SafeCast} from "./SafeCast.sol";

/**
 * @dev Standard math utilities missing in the Solidity language.
 */
library Math {
    enum Rounding {
        Floor, // Toward negative infinity
        Ceil, // Toward positive infinity
        Trunc, // Toward zero
        Expand // Away from zero
    }

    /**
     * @dev Return the 512-bit addition of two uint256.
     *
     * The result is stored in two 256 variables such that sum = high * 2²⁵⁶ + low.
     */
```

File 16 of 21 : SafeCast.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (utils/math/SafeCast.sol)
// This file was procedurally generated from scripts/generate/templates/SafeCast.js.

pragma solidity ^0.8.20;

/**
 * @dev Wrappers over Solidity's uintXX/intXX/bool casting operators with added overfl
 * checks.
 *
 * Downcasting from uint256/int256 in Solidity does not revert on overflow. This can
 * easily result in undesired exploitation or bugs, since developers usually
 * assume that overflows raise errors. `SafeCast` restores this intuition by
 * reverting the transaction when such 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 SafeCast {
    /**
     * @dev Value doesn't fit in an uint of `bits` size.
     */
    error SafeCastOverflowedUintDowncast(uint8 bits, uint256 value);

    /**
```

File 17 of 21 : SignedMath.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (utils/math/SignedMath.sol)

pragma solidity ^0.8.20;

import {SafeCast} from "./SafeCast.sol";

/**
 * @dev Standard signed math utilities missing in the Solidity language.
 */
library SignedMath {
    /**
     * @dev Branchless ternary evaluation for `a ? b : c`. Gas costs are constant.
     *
     * IMPORTANT: This function may reduce bytecode size and consume less gas when used
     * However, the compiler may optimize Solidity ternary operations (i.e. `a ? b : c`)
     * one branch when needed, making this function more expensive.
     */
    function ternary(bool condition, int256 a, int256 b) internal pure returns (int256)
        unchecked {
            // branchless ternary works because:
            // b ^ (a ^ b) == a
            // b ^ 0 == b
            return b ^ ((a ^ b) * int256(SafeCast.toUint(condition)));
        }
```

File 18 of 21 : Panic.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (utils/Panic.sol)

pragma solidity ^0.8.20;

/**
 * @dev Helper library for emitting standardized panic codes.
 *
 * ```solidity
 * contract Example {
 *      using Panic for uint256;
 *
 *      // Use any of the declared internal constants
```

```
 *      function foo() { Panic.GENERIC.panic(); }
 *
 *      // Alternatively
 *      function foo() { Panic.panic(Panic.GENERIC); }
 * }
 * ```
 *
 * Follows the list from https://github.com/ethereum/solidity/blob/v0.8.24/libsolutil/Er
 *
 * _Available since v5.1._
 */
```

File 19 of 21 : SlotDerivation.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.3.0) (utils/SlotDerivation.sol)
// This file was procedurally generated from scripts/generate/templates/SlotDerivation.

pragma solidity ^0.8.20;

/**
 * @dev Library for computing storage (and transient storage) locations from namespaces
 * corresponding to standard patterns. The derivation method for array and mapping matc
 * the solidity language / compiler.
 *
 * See https://docs.soliditylang.org/en/v0.8.20/internals/layout_in_storage.html#mappin
 *
 * Example usage:
 * ```solidity
 * contract Example {
 *     // Add the library methods
 *     using StorageSlot for bytes32;
 *     using SlotDerivation for bytes32;
 *
 *     // Declare a namespace
 *     string private constant _NAMESPACE = "<namespace>"; // eg. OpenZeppelin.Slot
 *
 *     function setValueInNamespace(uint256 key, address newValue) internal {
 *          NAMESPACE.erc7201Slot().deriveMapping(key).getAddressSlot().value = newValu
```

File 20 of 21 : StorageSlot.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.1.0) (utils/StorageSlot.sol)
// This file was procedurally generated from scripts/generate/templates/StorageSlot.js.

pragma solidity ^0.8.20;

/**
 * @dev Library for reading and writing primitive types to specific storage slots.
 *
 * Storage slots are often used to avoid storage conflict when dealing with upgradeable
 * This library helps with reading and writing to such slots without the need for inlin
 *
 * The functions in this library return Slot structs that contain a `value` member that
 *
 * Example usage to set ERC-1967 implementation slot:
 * ```solidity
 * contract ERC1967 {
 *     // Define the slot. Alternatively, use the SlotDerivation library to derive the
 *     bytes32 internal constant _IMPLEMENTATION_SLOT = 0x360894a13ba1a3210667c828492db
 *
 *     function _getImplementation() internal view returns (address) {
 *         return StorageSlot.getAddressSlot(_IMPLEMENTATION_SLOT).value;
 *     }
 *
 *     function  setImplementation(address newImplementation) internal {
```

File 21 of 21 : Strings.sol

```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v5.3.0) (utils/Strings.sol)

pragma solidity ^0.8.20;

import {Math} from "./math/Math.sol";
import {SafeCast} from "./math/SafeCast.sol";
import {SignedMath} from "./math/SignedMath.sol";

/**
 * @dev String operations.
 */
library Strings {
    using SafeCast for *;

    bytes16 private constant HEX_DIGITS = "0123456789abcdef";
    uint8 private constant ADDRESS_LENGTH = 20;
    uint256 private constant SPECIAL_CHARS_LOOKUP =
        (1 << 0x08) | // backspace
            (1 << 0x09) | // tab
            (1 << 0x0a) | // newline
            (1 << 0x0c) | // form feed
            (1 << 0x0d) | // carriage return
            (1 << 0x22) | // double quote
            (1 << 0x5c); // backslash
```

Settings

```json
{
  "evmVersion": "paris",
  "optimizer": {
    "enabled": false,
    "runs": 200
  },
  "outputSelection": {
    "*": {
      "*": [
        "evm.bytecode",
        "evm.deployedBytecode",
        "devdoc",
        "userdoc",
        "metadata",
        "abi"
      ]
    }
  },
  "libraries": {}
}
```

Contract Security Audit

- No Contract Security Audit Submitted - Submit Audit Here

Contract ABI API


```
[{"inputs":[{"internalType":"string","name":"_baseURI","type":"string"},
{"internalType":"address","name":"_owner","type":"address"}],"stateMutability":"nonpaya
ble","type":"constructor"},{"inputs":
[{"internalType":"address","name":"sender","type":"address"},
{"internalType":"uint256","name":"balance","type":"uint256"},
{"internalType":"uint256","name":"needed","type":"uint256"},
{"internalType":"uint256","name":"tokenId","type":"uint256"}],"name":"ERC1155Insufficie
ntBalance","type":"error"},{"inputs":
[{"internalType":"address","name":"approver","type":"address"}],"name":"ERC1155InvalidA
pprover","type":"error"},{"inputs":
```

Contract Creation Code

Decompile Bytecode | Switch to Opcodes View

```
608060405260405180602001604052806000815250600390816200002491906200bb6565b5060405180604
00160405280600d81526020017f4c4849204c65636365204e4654000000000000000000000000000000000000
0000815250600690816200006b919062000bb6565b506040518060400160405280600581526020017f4c484
94c450000000000000000000000000000000000000000000000000000000081525060079081620000b2919062
000bb6565b5034801562000c057600080fd5b506040516200577e3803806200577e8339818101604052810
190620000e6919062000e66565b8082620000f981620000860160201b60201c565b50600073ffffffffffffff
ffffffffffffffffffffffff168173ffffffffffffffffffffffffffffffffffffffff16036200016f576
0006040517f1e4fbdf700000000000000000000000000000000000000000000000000000000815260040162
00166919062000edd565b60405180910390fd5b6200018081620000876020 1b60201c565b5060008251116
20001c8576040517f08c379a000000000000000000000000000000000000000000000000000000000815260
```

Deployed Bytecode

```
0x608060405260043610610c15760003560e01c80636406c10c116100f757806395d89b411161009557806
3c48283ff11610064578063c48283ff1461067a578063e985e9c514610696578063f242432a146106d35780
63f2fde38b146106fc576101c1565b806395d89b41146105ac5780639d7f4ebf146105d7578063a22cb4651
```

Constructor Arguments (ABI-Encoded and is the last bytes of the Contract Creation Code above)

```
0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000
0f9909c6cd90566bd56621ee0cac42986ae334ea3000000000000000000000000000000000000000000000000
0000000000000000015697066733a2f2f516d4578616d706c65486173682f0000000000000000000000

-----Decoded View---------------
Arg [0] : _baseURI (string): ipfs://QmExampleHash/
Arg [1] : _owner (address): 0xf9909c6CD90566BD56621EE0cAc42986ae334Ea3

-----Encoded View---------------
4 Constructor Arguments found :
```

A contract address hosts a smart contract, which is a set of code stored on the blockchain that runs when predetermined conditions are met. Learn more about addresses in our Knowledge Base.