Mark5252h / **Drops.json**

Created 11 minutes ago

Embed ▾ <script src="https://gist. Download ZIP

Created using remix-ide: Realtime Ethereum Contract Compiler and Runtime. Load this file by pasting this gists URL or ID at https://remix.ethereum.org/#version=undefined&optimize=false&gist=

Drops.json


```json
{
	"deploy": {
		"VM:-": {
			"linkReferences": {},
			"autoDeployLib": true
		},
		"main:1": {
			"linkReferences": {},
			"autoDeployLib": true
		},
		"ropsten:3": {
			"linkReferences": {},
			"autoDeployLib": true
		},
		"rinkeby:4": {
			"linkReferences": {},
			"autoDeployLib": true
		},
		"kovan:42": {
			"linkReferences": {},
			"autoDeployLib": true
		},
		"görli:5": {
			"linkReferences": {},
			"autoDeployLib": true
		},
		"Custom": {
			"linkReferences": {},
			"autoDeployLib": true
		}
	},
	"data": {
		"bytecode": {
			"linkReferences": {},
			"object": "60806040523480156200001157600080fd5b5060405162001fb438038062001fb4833981810160405260208110156200003757 60
			"opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH3 0x11 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP
			"sourceMap": "653:1401:0:-;;;1362:689;8:9:-1;5:2;;;30:1;27;20:12;5:2;1362:689:0;;;;;;;;;;;;;13:2:-1;8:3;5:11;2:2;
		},
		"deployedBytecode": {
			"linkReferences": {},
			"object": "608060405260043610610 0a75760003560e01c806370a08231116100645780637 0a08231146102ea57806387bb70fb1461034f57
			"opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE LT PUSH2 0xA7 JUMPI PUSH1 0x0 CALLDATALOAD PUSH1 0x
			"sourceMap": "653:1401:0:-;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;2057:61
		},
		"gasEstimates": {
			"creation": {
				"codeDepositCost": "1464600",
				"executionCost": "infinite",
				"totalCost": "infinite"
			},
			"external": {
				"": "207",
				"allowance(address,address)": "831",
				"approve(address,uint256)": "infinite",
				"balanceOf(address)": "596",
				"burnCoins(uint256)": "infinite",
				"burnContract(string)": "infinite",
				"decimals()": "564",
				"mintCoins(uint256)": "infinite",
				"name()": "infinite",
				"symbol()": "infinite",
```

```
                        "totalSupply()": "480",
                        "transfer(address,uint256)": "infinite",
                        "transferFrom(address,address,uint256)": "infinite"
                }
        },
        "methodIdentifiers": {
                "allowance(address,address)": "dd62ed3e",
                "approve(address,uint256)": "095ea7b3",
                "balanceOf(address)": "70a08231",
                "burnCoins(uint256)": "05a8749d",
                "burnContract(string)": "87bb70fb",
                "decimals()": "313ce567",
                "mintCoins(uint256)": "a77b6efb",
                "name()": "06fdde03",
                "symbol()": "95d89b41",
                "totalSupply()": "18160ddd",
                "transfer(address,uint256)": "a9059cbb",
                "transferFrom(address,address,uint256)": "23b872dd"
        }
},
"abi": [
        {
                "constant": false,
                "inputs": [
                        {
                                "internalType": "uint256",
                                "name": "tokens_",
                                "type": "uint256"
                        }
                ],
                "name": "burnCoins",
                "outputs": [
                        {
                                "internalType": "uint256",
                                "name": "balance",
                                "type": "uint256"
                        }
                ],
                "payable": false,
                "stateMutability": "nonpayable",
                "type": "function"
        },
        {
                "constant": true,
                "inputs": [],
                "name": "name",
                "outputs": [
                        {
                                "internalType": "string",
                                "name": "",
                                "type": "string"
                        }
                ],
                "payable": false,
                "stateMutability": "view",
                "type": "function"
        },
        {
                "constant": false,
                "inputs": [
                        {
                                "internalType": "address",
                                "name": "delegate_",
                                "type": "address"
                        },
                        {
                                "internalType": "uint256",
                                "name": "tokens_",
                                "type": "uint256"
                        }
                ],
                "name": "approve",
                "outputs": [
```

```
                        {
                                "internalType": "bool",
                                "name": "sucess",
                                "type": "bool"
                        }
                ],
                "payable": false,
                "stateMutability": "nonpayable",
                "type": "function"
        },
        {
                "constant": true,
                "inputs": [],
                "name": "totalSupply",
                "outputs": [
                        {
                                "internalType": "uint256",
                                "name": "supply",
                                "type": "uint256"
                        }
                ],
                "payable": false,
                "stateMutability": "view",
                "type": "function"
        },
        {
                "constant": false,
                "inputs": [
                        {
                                "internalType": "address",
                                "name": "owner_",
                                "type": "address"
                        },
                        {
                                "internalType": "address",
                                "name": "receiver_",
                                "type": "address"
                        },
                        {
                                "internalType": "uint256",
                                "name": "tokens_",
                                "type": "uint256"
                        }
                ],
                "name": "transferFrom",
                "outputs": [
                        {
                                "internalType": "bool",
                                "name": "sucess",
                                "type": "bool"
                        }
                ],
                "payable": false,
                "stateMutability": "nonpayable",
                "type": "function"
        },
        {
                "constant": true,
                "inputs": [],
                "name": "decimals",
                "outputs": [
                        {
                                "internalType": "uint8",
                                "name": "",
                                "type": "uint8"
                        }
                ],
                "payable": false,
                "stateMutability": "view",
                "type": "function"
        },
        {
                "constant": true,
```

```
                        "inputs": [
                                {
                                        "internalType": "address",
                                        "name": "owner_",
                                        "type": "address"
                                }
                        ],
                        "name": "balanceOf",
                        "outputs": [
                                {
                                        "internalType": "uint256",
                                        "name": "balance",
                                        "type": "uint256"
                                }
                        ],
                        "payable": false,
                        "stateMutability": "view",
                        "type": "function"
                },
                {
                        "constant": false,
                        "inputs": [
                                {
                                        "internalType": "string",
                                        "name": "killCode_",
                                        "type": "string"
                                }
                        ],
                        "name": "burnContract",
                        "outputs": [],
                        "payable": false,
                        "stateMutability": "nonpayable",
                        "type": "function"
                },
                {
                        "constant": true,
                        "inputs": [],
                        "name": "symbol",
                        "outputs": [
                                {
                                        "internalType": "string",
                                        "name": "",
                                        "type": "string"
                                }
                        ],
                        "payable": false,
                        "stateMutability": "view",
                        "type": "function"
                },
                {
                        "constant": false,
                        "inputs": [
                                {
                                        "internalType": "uint256",
                                        "name": "tokens_",
                                        "type": "uint256"
                                }
                        ],
                        "name": "mintCoins",
                        "outputs": [
                                {
                                        "internalType": "uint256",
                                        "name": "balance",
                                        "type": "uint256"
                                }
                        ],
                        "payable": false,
                        "stateMutability": "nonpayable",
                        "type": "function"
                },
                {
                        "constant": false,
                        "inputs": [
```

```
                        {
                                "internalType": "address",
                                "name": "receiver_",
                                "type": "address"
                        },
                        {
                                "internalType": "uint256",
                                "name": "tokens_",
                                "type": "uint256"
                        }
                ],
                "name": "transfer",
                "outputs": [
                        {
                                "internalType": "bool",
                                "name": "sucess",
                                "type": "bool"
                        }
                ],
                "payable": false,
                "stateMutability": "nonpayable",
                "type": "function"
        },
        {
                "constant": true,
                "inputs": [
                        {
                                "internalType": "address",
                                "name": "owner_",
                                "type": "address"
                        },
                        {
                                "internalType": "address",
                                "name": "delegate_",
                                "type": "address"
                        }
                ],
                "name": "allowance",
                "outputs": [
                        {
                                "internalType": "uint256",
                                "name": "remaining",
                                "type": "uint256"
                        }
                ],
                "payable": false,
                "stateMutability": "view",
                "type": "function"
        },
        {
                "inputs": [
                        {
                                "internalType": "uint256",
                                "name": "initialMint_",
                                "type": "uint256"
                        }
                ],
                "payable": false,
                "stateMutability": "nonpayable",
                "type": "constructor"
        },
        {
                "payable": true,
                "stateMutability": "payable",
                "type": "fallback"
        },
        {
                "anonymous": false,
                "inputs": [
                        {
                                "indexed": true,
                                "internalType": "address",
                                "name": "owner_",
```

```
                                        "type": "address"
                                },
                                {
                                        "indexed": true,
                                        "internalType": "address",
                                        "name": "receiver_",
                                        "type": "address"
                                },
                                {
                                        "indexed": false,
                                        "internalType": "uint256",
                                        "name": "tokens_",
                                        "type": "uint256"
                                }
                        ],
                        "name": "Transfer",
                        "type": "event"
                },
                {
                        "anonymous": false,
                        "inputs": [
                                {
                                        "indexed": true,
                                        "internalType": "address",
                                        "name": "owner_",
                                        "type": "address"
                                },
                                {
                                        "indexed": true,
                                        "internalType": "address",
                                        "name": "delegate_",
                                        "type": "address"
                                },
                                {
                                        "indexed": false,
                                        "internalType": "uint256",
                                        "name": "tokens_",
                                        "type": "uint256"
                                }
                        ],
                        "name": "Approval",
                        "type": "event"
                }
        ]
}
```

Drops.sol

```
pragma solidity ^0.5.1;
import "./ERC20_Token.sol";
/**
 * @author  Mark R Rogers
 *
 * LICENSE
 *
 * This program is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * DESCRIPTION
 *
 * The Drops contract is built upon the ERC20_Token contract which
 * is in turn build upon the genesis contract
 *
 **/

 /**
 * @title Drops (DRP)
 * @dev Drops crypto currency smart contract based upon the ERC20 standart token
 *
 * Drops (DRP) Smart Contract v1.0.1 08.09.2019,  Mark R Rogers, 2019.
 *
```

```
 **/
contract Drops is ERC20_Token {                                                 // Build on ERC20 standard contract
    // Initalise variables and data
    string constant TOKEN_NAME = "Drops";                                       // Token description
    string constant TOKEN_SYMBOL = "DRP";                                       // Token symbol
    uint8  constant TOKEN_DECIMALS = 8;                                         // Token decimals

    /** Initial contract deployment setup
     * @param initialMint_  The amount of coins to create
     *
     * note:
     * This is only run once when the contract is deployed and is not contained in memory bytecode
    **/
    constructor(uint256 initialMint_) public {
        name = TOKEN_NAME;                                                      // Set description
        symbol = TOKEN_SYMBOL;                                                  // Set symbol
        decimals = TOKEN_DECIMALS;                                              // Set decimals
        coinOwner = msg.sender;                                                 // Set coin owner identity
        coinSupply = initialMint_.toklets(TOKEN_DECIMALS);                      // Set total supply in droplets
        balances[msg.sender] = coinSupply;                                      // Set owners balance
    }
}
```

ERC20_Token.sol

```
pragma solidity ^0.5.1;
import "./Genesis.sol";
/**
 * @author  Mark R Rogers
 *
 *
 * LICENSE
 *
 * This program is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * DESCRIPTION
 *
 * The ERC20_Token contract is built upon the genesis contract.
 **/

 /**
 * @title ERC20 Standard Token
 *
 * Inherits (Genesis::) events, modifiers & data from the genesis contract which complies with the ERC20 token Standard.
 * Inizalizes genesis functions & extra added functions
 *
 **/
contract ERC20_Token is Genesis {                                                 // Build on genesis contract
    // Initalise global constants
    string constant ERR_INSUFFICENT_BALANCE = "Insufficent amount of DRP";      // Error message 102
    string constant ERR_INVALID_DELEGATE    = "Invalid delegate address";       // Error message 103
    string constant ERR_ALLOWANCE_EXCEEDED  = "Allowance exceeded!";            // Error message 104
    string constant ERR_INVALID_KILL_CODE   = "Invalid kill code!";             // Error message 105
    string constant KILL_CODE               = "K-C102-473";                     // WARNING! Contracts kill code

    /** Create new tokens
     * @param tokens_   Number of new tokens to create
     *
     * @return The total supply of tokens
    **/
    function mintCoins(uint tokens_) ownerOnly public returns (uint balance) {
        tokens_ = tokens_.toklets(decimals);                                    // Convert tokens to toklets
        coinSupply = coinSupply.add(tokens_);                                   // Create new tokens
        balances[coinOwner] = balances[coinOwner].add(tokens_);                 // Update owners balace
      return coinSupply;
    }

    /** Destroy tokens
     * @param tokens_   Number of tokens to destroy
     *
     * @return The total supply of tokens
```

```
    **/
    function burnCoins(uint tokens_) ownerOnly public returns (uint balance) {      // Restricted to owner only
        tokens_ = tokens_.toklets(decimals);                                        // Convert tokens to toklets
        if (valid(tokens_ <= balances[coinOwner], 102)) {                           // Check enough tokens available
            coinSupply = coinSupply.sub(tokens_);                                   // Decrease total coin supply
            balances[coinOwner] = balances[coinOwner].sub(tokens_);                 // Update owners token balance
          return coinSupply;
        }
    }

    // Genesis:: Transfer tokens to receiver
    function transfer(address receiver_,
                    uint tokens_) public returns (bool sucess) {
      super.transfer(receiver_, tokens_);
        if (valid(tokens_ <= balances[msg.sender] &&                                // Check enough tokens available
            tokens_ > 0, 102)) {                                                    // and amount greater than zero
            balances[msg.sender] = balances[msg.sender].sub(tokens_);               // Decrease senders token balance
            balances[receiver_] = balances[receiver_].add(tokens_);                 // Increase receivers token balance
            emit Transfer(msg.sender, receiver_, tokens_);                          // Transfer tokens
          return true;
        }
    }

    // Genesis:: Approve token allowence for delegate
    function approve(address delegate_,
                   uint tokens_) public returns (bool sucess) {
      super.approve(delegate_, tokens_);
        if (valid(delegate_ != msg.sender, 103)) {                                  // Check not delegating to yourself
            if (tokens_ > coinSupply) { tokens_ = coinSupply; }                     // Limit allowance to total supply
            allowed[msg.sender][delegate_] = tokens_;                               // Update token allowence
            emit Approval(msg.sender, delegate_, tokens_);                          // Approve token allowance
          return true;
        }
    }

    // Genesis:: Transfer token from delegated address
    function transferFrom(address owner_, address receiver_,
                        uint tokens_) public returns (bool sucess) {
      super.transferFrom(owner_ , receiver_, tokens_);
        if (valid(tokens_ > 0 && tokens_ <= balances[owner_], 102) &&               // Check amount greater than zero and enough tokens ava
            valid(tokens_ <= allowed[owner_][msg.sender], 104)) {                   // Make sure smount is equal or less than token allowan
            balances[owner_] = balances[owner_].sub(tokens_);                       // Decrease owner of tokens balance
            allowed[owner_][msg.sender] = allowed[owner_][msg.sender].sub(tokens_); // Decrease senders tokens allowance
            balances[receiver_] = balances[receiver_].add(tokens_);                 // Increase receivers tokens balance
            emit Transfer(owner_, receiver_, tokens_);                              // Transfer tokens from the owner to the receiver
        return true;
        }
    }

    /** (internal) Validation of expressions and error handling
     * @param valid_    Expression to varify
     * @param errorID_  The error handled on failed varification
     *
     * @return If the expression was valid
    **/
    function valid(bool valid_, uint errorID_) internal pure returns (bool) {       // Check for fatal errors
        if (errorID_ == 101) {require(valid_, ERR_PERMISSION_DENIED);}              // Calling address doesn't have permission
          else if (errorID_ == 102) {require(valid_, ERR_INSUFFICENT_BALANCE);}     // Cancel trasaction due to insufficent value
          else if (errorID_ == 103) {require(valid_, ERR_INVALID_DELEGATE);}        // Cannot delegate to address
          else if (errorID_ == 104) {require(valid_, ERR_ALLOWANCE_EXCEEDED);}      // Cancel trasaction due to insufficent value
          else if (errorID_ == 105) {require(valid_, ERR_INVALID_KILL_CODE);}       // Cancel trasaction due to insufficent value
          else if (errorID_ == 100) {require (valid_);}                             // Check if required?
        return valid_;
    }

    /**  Terminates contract
     * @param killCode_ The contracts kill code
     *
     * note:
     * ! WARNING ! CONFIRM NOTHING ELSE NEEDS THIS CONTRACT BEFORE BURNING IT!
    **/
    function burnContract(string memory killCode_) ownerOnly public {
        if (valid((keccak256(abi.encodePacked(killCode_)) ==
```

```
                keccak256(abi.encodePacked(KILL_CODE))), 105))                  // Authenticate kill code
                { selfdestruct(address(0)); }                                   // Kill contract
    }
}
```

Genesis.sol

```
pragma solidity ^0.5.1;

/**
 * @author  Mark R Rogers
 * @version v1.0.1
 *
 * LICENSE
 *
 * This program is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * DESCRIPTION
 *
 * The Genesis contract is the base contract for all other contracts.
 **/

 /**
 * @title SafeMath Library
 * @dev (internal) Library for math operations which checks logic and throws on error
 *
 **/
library SafeMath {
    /** Converts tokens into toklets
    * @param numA_  The amount of tokens
    * @param numB_  The number of decimal places
    *
    * @return The product
    **/
    function toklets(uint256 numA_, uint8 numB_) internal pure returns (uint256) {
        uint256 numD_ = 10**uint256(numB_);
        uint256 numC_ = numA_ * numD_;
        require(numA_ > 0 && numC_ / numA_ == numD_, "Invalid amount of tokens");
      return numC_;
    }

    /** Multipy unsigned integers
    * @param numA_  The LHS factor
    * @param numB_  The RHS factor
    *
    * @return The product
    **/
    function mul(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
        uint256 numC_ = numA_ * numB_;
        assert(numA_ == 0 || numC_ / numA_ == numB_);
      return numC_;
    }

    /** Divide unsigned integers
    * @param numA_  The LHS factor
    * @param numB_  The RHS factor
    *
    * @return The product
    **/
    function div(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
        uint256 numC_ = numA_ / numB_;                                          // Solidity automatically throws when divi
      return numC_;
    }

    /** Subtract unsigned integers
    * @param numA_  The Minuend
    * @param numB_  The Subtrahend
    *
    * @return The Difference
    **/
```

```
    function sub(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
        assert(numB_ <= numA_);
      return numA_ - numB_;
    }

    /** Add unsigned integer values and check logic
    * @param numA_  The LHS addend
    * @param numB_  The RHS addend
    *
    * @return The sum
    **/
    function add(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
        uint256 numC_ = numA_ + numB_;
        assert(numC_ >= numA_);
      return numC_;
    }
}

/**
 *
 * @title Genesis Contract
 *
 * Initializes events, modifiers & data in the contract and defines default functionality
 * that follows the ERC20 standard token format
 *
 **/
contract Genesis {
    using SafeMath for uint256;                                                 // Use SafeMath library to test the logic of uint256 ca

    // Initalise contract global constants
    string constant ERR_PERMISSION_DENIED   = "Permission denied!";             // Error message 101

    // Initalise token information
    string public name;                                                         // Token Name
    string public symbol;                                                       // Token Symbol
    uint8  public decimals;                                                     // Token decimals (droplets)
    address coinOwner ;                                                         // Token owners address
    uint256 coinSupply;                                                         // Total token supply
    mapping(address => uint256) balances;                                       // Token balance state
    mapping(address => mapping (address => uint256)) allowed;                   // Token allowance state

    // Owner privelages only
    modifier ownerOnly() {
        require(msg.sender == coinOwner, ERR_PERMISSION_DENIED) ;
        _;
    }

    // Transfer tokens
    event Transfer(address indexed owner_, address indexed receiver_, uint256 tokens_);

    // Approve token allowances
    event Approval(address indexed owner_, address indexed delegate_, uint256 tokens_);

    // Fallback function handles unidentified calls and allows contract to receive payments
    function() payable external { }

    // @return total supply of tokens
    function totalSupply() external view returns (uint256 supply) { return coinSupply; }

    /** Gets the balance of tokens for an address
     * @param owner_    The address owning The tokens
     *
     * @return The balance of tokens
    **/
    function balanceOf(address owner_) external view returns (uint balance) { return balances[owner_]; }

    /** Transfer tokens to an address
    * @param receiver_  The address of the recipient
    * @param tokens_    The amount of token to be transferred
    *
    * @return Whether the approval was successful or not
    **/
    function transfer(address receiver_, uint tokens_) public returns (bool sucess) {}
```

```
139
140     /** Approve an address an allowance of owners tokens
141      * @param delegate_ The address allowed to transfer the tokens
142      * @param tokens_   The allowance of tokens allowed to transfer
143      *
144      * @return Whether the address was approved or not
145      */
146     function approve(address delegate_, uint tokens_) public returns (bool sucess) {}
147
148     /** Gets the allowance of owners tokens for an address
149      * @param owner_    The address owning the tokens
150      * @param delegate_ The address allowed to transfer the tokens
151      *
152      * @return The remaining allowance
153     **/
154     function allowance(address owner_, address delegate_) external view returns (uint remaining) { return allowed[owner_][delegate_]; }
155
156     /** Transfer tokens from allowance at address owning them to an address
157      * @param owner_    The address owning the tokens
158      * @param receiver_ The address of the recipient
159      * @param tokens_   The amount of token to be transferred
160      *
161      * @return Whether the approval was successful or not
162     **/
163     function transferFrom(address owner_, address receiver_, uint tokens_) public returns (bool sucess) { }
164 }
```