



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

```
1  {
2      "deploy": {
3          "VM:-": {
4              "linkReferences": {},
5              "autoDeployLib": true
6          },
7          "main:1": {
8              "linkReferences": {},
9              "autoDeployLib": true
10         },
11         "ropsten:3": {
12             "linkReferences": {},
13             "autoDeployLib": true
14         },
15         "rinkeby:4": {
16             "linkReferences": {},
17             "autoDeployLib": true
18         },
19         "kovan:42": {
20             "linkReferences": {},
21             "autoDeployLib": true
22         },
23         "görli:5": {
24             "linkReferences": {},
25             "autoDeployLib": true
26         },
27         "Custom": {
28             "linkReferences": {},
29             "autoDeployLib": true
30         }
31     },
32     "data": {
33         "bytecode": {
34             "linkReferences": {},
35             "object": "60806040523480156200001157600080fd5b5060405162001fb438038062001fb483398181016040526020811015620000375760
36             "opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH3 0x11 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP
37             "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;
38         },
39         "deployedBytecode": {
40             "linkReferences": {},
41             "object": "608060405260436106100a75760003560e01c806370a082311161006457806370a08231146102ea57806387bb70fb1461034f57
42             "opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE LT PUSH2 0xa7 JUMPI PUSH1 0x0 CALLDATALOAD PUSH1 0x
43             "sourceMap": "653:1401:0:-;:::;2057:61
44         },
45         "gasEstimates": {
46             "creation": {
47                 "codeDepositCost": "1464600",
48                 "executionCost": "infinite",
49                 "totalCost": "infinite"
50             },
51             "external": {
52                 "": "207",
53                 "allowance(address,address)": "831",
54                 "approve(address,uint256)": "infinite",
55                 "balanceOf(address)": "596",
56                 "burnCoins(uint256)": "infinite",
57                 "burnContract(string)": "infinite",
58                 "decimals()": "564",
59                 "mintCoins(uint256)": "infinite",
60                 "name()": "infinite",
61                 "symbol()": "infinite",
```

```

62         "totalSupply()": "480",
63         "transfer(address,uint256)": "infinite",
64         "transferFrom(address,address,uint256)": "infinite"
65     }
66 },
67 "methodIdentifiers": {
68     "allowance(address,address)": "dd62ed3e",
69     "approve(address,uint256)": "095ea7b3",
70     "balanceOf(address)": "70a08231",
71     "burnCoins(uint256)": "05a8749d",
72     "burnContract(string)": "87bb70fb",
73     "decimals()": "313ce567",
74     "mintCoins(uint256)": "a77b6efb",
75     "name()": "06fdde03",
76     "symbol()": "95d89b41",
77     "totalSupply()": "18160ddd",
78     "transfer(address,uint256)": "a9059cbb",
79     "transferFrom(address,address,uint256)": "23b872dd"
80 }
81 },
82 "abi": [
83     {
84         "constant": false,
85         "inputs": [
86             {
87                 "internalType": "uint256",
88                 "name": "tokens_",
89                 "type": "uint256"
90             }
91         ],
92         "name": "burnCoins",
93         "outputs": [
94             {
95                 "internalType": "uint256",
96                 "name": "balance",
97                 "type": "uint256"
98             }
99         ],
100         "payable": false,
101         "stateMutability": "nonpayable",
102         "type": "function"
103     },
104     {
105         "constant": true,
106         "inputs": [],
107         "name": "name",
108         "outputs": [
109             {
110                 "internalType": "string",
111                 "name": "",
112                 "type": "string"
113             }
114         ],
115         "payable": false,
116         "stateMutability": "view",
117         "type": "function"
118     },
119     {
120         "constant": false,
121         "inputs": [
122             {
123                 "internalType": "address",
124                 "name": "delegate_",
125                 "type": "address"
126             },
127             {
128                 "internalType": "uint256",
129                 "name": "tokens_",
130                 "type": "uint256"
131             }
132         ],
133         "name": "approve",
134         "outputs": [

```

```

135         {
136             "internalType": "bool",
137             "name": "sucess",
138             "type": "bool"
139         }
140     ],
141     "payable": false,
142     "stateMutability": "nonpayable",
143     "type": "function"
144 },
145 {
146     "constant": true,
147     "inputs": [],
148     "name": "totalSupply",
149     "outputs": [
150         {
151             "internalType": "uint256",
152             "name": "supply",
153             "type": "uint256"
154         }
155     ],
156     "payable": false,
157     "stateMutability": "view",
158     "type": "function"
159 },
160 {
161     "constant": false,
162     "inputs": [
163         {
164             "internalType": "address",
165             "name": "owner_",
166             "type": "address"
167         },
168         {
169             "internalType": "address",
170             "name": "receiver_",
171             "type": "address"
172         },
173         {
174             "internalType": "uint256",
175             "name": "tokens_",
176             "type": "uint256"
177         }
178     ],
179     "name": "transferFrom",
180     "outputs": [
181         {
182             "internalType": "bool",
183             "name": "sucess",
184             "type": "bool"
185         }
186     ],
187     "payable": false,
188     "stateMutability": "nonpayable",
189     "type": "function"
190 },
191 {
192     "constant": true,
193     "inputs": [],
194     "name": "decimals",
195     "outputs": [
196         {
197             "internalType": "uint8",
198             "name": "",
199             "type": "uint8"
200         }
201     ],
202     "payable": false,
203     "stateMutability": "view",
204     "type": "function"
205 },
206 {
207     "constant": true,

```

```

208         "inputs": [
209             {
210                 "internalType": "address",
211                 "name": "owner_",
212                 "type": "address"
213             }
214         ],
215         "name": "balanceOf",
216         "outputs": [
217             {
218                 "internalType": "uint256",
219                 "name": "balance",
220                 "type": "uint256"
221             }
222         ],
223         "payable": false,
224         "stateMutability": "view",
225         "type": "function"
226     },
227     {
228         "constant": false,
229         "inputs": [
230             {
231                 "internalType": "string",
232                 "name": "killCode_",
233                 "type": "string"
234             }
235         ],
236         "name": "burnContract",
237         "outputs": [],
238         "payable": false,
239         "stateMutability": "nonpayable",
240         "type": "function"
241     },
242     {
243         "constant": true,
244         "inputs": [],
245         "name": "symbol",
246         "outputs": [
247             {
248                 "internalType": "string",
249                 "name": "",
250                 "type": "string"
251             }
252         ],
253         "payable": false,
254         "stateMutability": "view",
255         "type": "function"
256     },
257     {
258         "constant": false,
259         "inputs": [
260             {
261                 "internalType": "uint256",
262                 "name": "tokens_",
263                 "type": "uint256"
264             }
265         ],
266         "name": "mintCoins",
267         "outputs": [
268             {
269                 "internalType": "uint256",
270                 "name": "balance",
271                 "type": "uint256"
272             }
273         ],
274         "payable": false,
275         "stateMutability": "nonpayable",
276         "type": "function"
277     },
278     {
279         "constant": false,
280         "inputs": [

```

```

281         {
282             "internalType": "address",
283             "name": "receiver_",
284             "type": "address"
285         },
286         {
287             "internalType": "uint256",
288             "name": "tokens_",
289             "type": "uint256"
290         }
291     ],
292     "name": "transfer",
293     "outputs": [
294         {
295             "internalType": "bool",
296             "name": "sucess",
297             "type": "bool"
298         }
299     ],
300     "payable": false,
301     "stateMutability": "nonpayable",
302     "type": "function"
303 },
304 {
305     "constant": true,
306     "inputs": [
307         {
308             "internalType": "address",
309             "name": "owner_",
310             "type": "address"
311         },
312         {
313             "internalType": "address",
314             "name": "delegate_",
315             "type": "address"
316         }
317     ],
318     "name": "allowance",
319     "outputs": [
320         {
321             "internalType": "uint256",
322             "name": "remaining",
323             "type": "uint256"
324         }
325     ],
326     "payable": false,
327     "stateMutability": "view",
328     "type": "function"
329 },
330 {
331     "inputs": [
332         {
333             "internalType": "uint256",
334             "name": "initialMint_",
335             "type": "uint256"
336         }
337     ],
338     "payable": false,
339     "stateMutability": "nonpayable",
340     "type": "constructor"
341 },
342 {
343     "payable": true,
344     "stateMutability": "payable",
345     "type": "fallback"
346 },
347 {
348     "anonymous": false,
349     "inputs": [
350         {
351             "indexed": true,
352             "internalType": "address",
353             "name": "owner_",

```

```

354         "type": "address"
355     },
356     {
357         "indexed": true,
358         "internalType": "address",
359         "name": "receiver_",
360         "type": "address"
361     },
362     {
363         "indexed": false,
364         "internalType": "uint256",
365         "name": "tokens_",
366         "type": "uint256"
367     }
368 ],
369 "name": "Transfer",
370 "type": "event"
371 },
372 {
373     "anonymous": false,
374     "inputs": [
375         {
376             "indexed": true,
377             "internalType": "address",
378             "name": "owner_",
379             "type": "address"
380         },
381         {
382             "indexed": true,
383             "internalType": "address",
384             "name": "delegate_",
385             "type": "address"
386         },
387         {
388             "indexed": false,
389             "internalType": "uint256",
390             "name": "tokens_",
391             "type": "uint256"
392         }
393     ],
394     "name": "Approval",
395     "type": "event"
396 }
397 ]
398 }

```

Drops.sol

```

1  pragma solidity ^0.5.1;
2  import "./ERC20_Token.sol";
3  /**
4   * @author Mark R Rogers
5   *
6   * LICENSE
7   *
8   * This program is distributed in the hope that it will be useful, but
9   * WITHOUT ANY WARRANTY; without even the implied warranty of
10  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
11  *
12  * DESCRIPTION
13  *
14  * The Drops contract is built upon the ERC20_Token contract which
15  * is in turn build upon the genesis contract
16  *
17  */
18
19 /**
20  * @title Drops (DRP)
21  * @dev Drops crypto currency smart contract based upon the ERC20 standart token
22  *
23  * Drops (DRP) Smart Contract v1.0.1 08.09.2019, Mark R Rogers, 2019.
24  *

```

```

25  /**
26  contract Drops is ERC20_Token {                                // Build on ERC20 standard contract
27      // Initialise variables and data
28      string constant TOKEN_NAME = "Drops";                      // Token description
29      string constant TOKEN_SYMBOL = "DRP";                      // Token symbol
30      uint8 constant TOKEN_DECIMALS = 8;                         // Token decimals
31
32      /** Initial contract deployment setup
33       * @param initialMint_ The amount of coins to create
34       *
35       * note:
36       * This is only run once when the contract is deployed and is not contained in memory bytecode
37      */
38      constructor(uint256 initialMint_) public {
39          name = TOKEN_NAME;                                       // Set description
40          symbol = TOKEN_SYMBOL;                                   // Set symbol
41          decimals = TOKEN_DECIMALS;                              // Set decimals
42          coinOwner = msg.sender;                                  // Set coin owner identity
43          coinSupply = initialMint_.toklets(TOKEN_DECIMALS);      // Set total supply in droplets
44          balances[msg.sender] = coinSupply;                      // Set owners balance
45      }
46  }

```

ERC20_Token.sol

```

1  pragma solidity ^0.5.1;
2  import "./Genesis.sol";
3  /**
4   * @author Mark R Rogers
5   *
6   *
7   * LICENSE
8   *
9   * This program is distributed in the hope that it will be useful, but
10  * WITHOUT ANY WARRANTY; without even the implied warranty of
11  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
12  *
13  * DESCRIPTION
14  *
15  * The ERC20_Token contract is built upon the genesis contract.
16  */
17
18  /**
19  * @title ERC20 Standard Token
20  *
21  * Inherits (Genesis::) events, modifiers & data from the genesis contract which complies with the ERC20 token Standard.
22  * Initalizes genesis functions & extra added functions
23  *
24  */
25  contract ERC20_Token is Genesis {                                // Build on genesis contract
26      // Initialise global constants
27      string constant ERR_INSUFFICIENT_BALANCE = "Insufficient amount of DRP"; // Error message 102
28      string constant ERR_INVALID_DELEGATE = "Invalid delegate address";      // Error message 103
29      string constant ERR_ALLOWANCE_EXCEEDED = "Allowance exceeded!";        // Error message 104
30      string constant ERR_INVALID_KILL_CODE = "Invalid kill code!";           // Error message 105
31      string constant KILL_CODE = "K-C102-473";                             // WARNING! Contracts kill code
32
33      /** Create new tokens
34       * @param tokens_ Number of new tokens to create
35       *
36       * @return The total supply of tokens
37      */
38      function mintCoins(uint tokens_) ownerOnly public returns (uint balance) {
39          tokens_ = tokens_.toklets(decimals); // Convert tokens to toklets
40          coinSupply = coinSupply.add(tokens_); // Create new tokens
41          balances[coinOwner] = balances[coinOwner].add(tokens_); // Update owners balace
42          return coinSupply;
43      }
44
45      /** Destroy tokens
46       * @param tokens_ Number of tokens to destroy
47       *
48       * @return The total supply of tokens

```

```

49  */
50  function burnCoins(uint tokens_) ownerOnly public returns (uint balance) {    // Restricted to owner only
51      tokens_ = tokens_.toklets(decimals);    // Convert tokens to toklets
52      if (valid(tokens_ <= balances[coinOwner], 102)) {    // Check enough tokens available
53          coinSupply = coinSupply.sub(tokens_);    // Decrease total coin supply
54          balances[coinOwner] = balances[coinOwner].sub(tokens_);    // Update owners token balance
55          return coinSupply;
56      }
57  }
58
59  // Genesis:: Transfer tokens to receiver
60  function transfer(address receiver_,
61      uint tokens_) public returns (bool success) {
62      super.transfer(receiver_, tokens_);
63      if (valid(tokens_ <= balances[msg.sender] &&    // Check enough tokens available
64          tokens_ > 0, 102)) {    // and amount greater than zero
65          balances[msg.sender] = balances[msg.sender].sub(tokens_);    // Decrease senders token balance
66          balances[receiver_] = balances[receiver_].add(tokens_);    // Increase receivers token balance
67          emit Transfer(msg.sender, receiver_, tokens_);    // Transfer tokens
68          return true;
69      }
70  }
71
72  // Genesis:: Approve token allowance for delegate
73  function approve(address delegate_,
74      uint tokens_) public returns (bool success) {
75      super.approve(delegate_, tokens_);
76      if (valid(delegate_ != msg.sender, 103)) {    // Check not delegating to yourself
77          if (tokens_ > coinSupply) { tokens_ = coinSupply; }    // Limit allowance to total supply
78          allowed[msg.sender][delegate_] = tokens_;    // Update token allowance
79          emit Approval(msg.sender, delegate_, tokens_);    // Approve token allowance
80          return true;
81      }
82  }
83
84  // Genesis:: Transfer token from delegated address
85  function transferFrom(address owner_, address receiver_,
86      uint tokens_) public returns (bool success) {
87      super.transferFrom(owner_, receiver_, tokens_);
88      if (valid(tokens_ > 0 && tokens_ <= balances[owner_], 102) &&    // Check amount greater than zero and enough tokens ava
89          valid(tokens_ <= allowed[owner_][msg.sender], 104)) {    // Make sure smount is equal or less than token allowan
90          balances[owner_] = balances[owner_].sub(tokens_);    // Decrease owner of tokens balance
91          allowed[owner_][msg.sender] = allowed[owner_][msg.sender].sub(tokens_);    // Decrease senders tokens allowance
92          balances[receiver_] = balances[receiver_].add(tokens_);    // Increase receivers tokens balance
93          emit Transfer(owner_, receiver_, tokens_);    // Transfer tokens from the owner to the receiver
94          return true;
95      }
96  }
97
98  /** (internal) Validation of expressions and error handling
99   * @param valid_ Expression to varify
100  * @param errorID_ The error handled on failed varification
101  *
102  * @return If the expression was valid
103  */
104  function valid(bool valid_, uint errorID_) internal pure returns (bool) {    // Check for fatal errors
105      if (errorID_ == 101) {require(valid_, ERR_PERMISSION_DENIED);}    // Calling address doesn't have permission
106      else if (errorID_ == 102) {require(valid_, ERR_INSUFFICIENT_BALANCE);}    // Cancel trasaction due to insufficient value
107      else if (errorID_ == 103) {require(valid_, ERR_INVALID_DELEGATE);}    // Cannot delegate to address
108      else if (errorID_ == 104) {require(valid_, ERR_ALLOWANCE_EXCEEDED);}    // Cancel trasaction due to insufficient value
109      else if (errorID_ == 105) {require(valid_, ERR_INVALID_KILL_CODE);}    // Cancel trasaction due to insufficient value
110      else if (errorID_ == 100) {require (valid_);}    // Check if required?
111      return valid_;
112  }
113
114  /** Terminates contract
115   * @param killCode_ The contracts kill code
116   *
117   * note:
118   * ! WARNING ! CONFIRM NOTHING ELSE NEEDS THIS CONTRACT BEFORE BURNING IT!
119  */
120  function burnContract(string memory killCode_) ownerOnly public {
121      if (valid((keccak256(abi.encodePacked(killCode_)) ==

```



```

122         keccak256(abi.encodePacked(KILL_CODE))), 105))           // Authenticate kill code
123         { selfdestruct(address(0)); }                             // Kill contract
124     }
125 }

```

Genesis.sol

```

1  pragma solidity ^0.5.1;
2
3  /**
4   * @author Mark R Rogers
5   * @version v1.0.1
6   *
7   * LICENSE
8   *
9   * This program is distributed in the hope that it will be useful, but
10  * WITHOUT ANY WARRANTY; without even the implied warranty of
11  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
12  *
13  * DESCRIPTION
14  *
15  * The Genesis contract is the base contract for all other contracts.
16  */
17
18  /**
19  * @title SafeMath Library
20  * @dev (internal) Library for math operations which checks logic and throws on error
21  *
22  */
23  library SafeMath {
24      /** Converts tokens into toklets
25       * @param numA_ The amount of tokens
26       * @param numB_ The number of decimal places
27       *
28       * @return The product
29       */
30      function toklets(uint256 numA_, uint8 numB_) internal pure returns (uint256) {
31          uint256 numD_ = 10**uint256(numB_);
32          uint256 numC_ = numA_ * numD_;
33          require(numA_ > 0 && numC_ / numA_ == numD_, "Invalid amount of tokens");
34          return numC_;
35      }
36
37      /** Multiply unsigned integers
38       * @param numA_ The LHS factor
39       * @param numB_ The RHS factor
40       *
41       * @return The product
42       */
43      function mul(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
44          uint256 numC_ = numA_ * numB_;
45          assert(numA_ == 0 || numC_ / numA_ == numB_);
46          return numC_;
47      }
48
49      /** Divide unsigned integers
50       * @param numA_ The LHS factor
51       * @param numB_ The RHS factor
52       *
53       * @return The product
54       */
55      function div(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
56          uint256 numC_ = numA_ / numB_; // Solidity automatically throws when divi
57          return numC_;
58      }
59
60      /** Subtract unsigned integers
61       * @param numA_ The Minuend
62       * @param numB_ The Subtrahend
63       *
64       * @return The Difference
65       */

```

```

66     function sub(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
67         assert(numB_ <= numA_);
68         return numA_ - numB_;
69     }
70
71     /** Add unsigned integer values and check logic
72     * @param numA_ The LHS addend
73     * @param numB_ The RHS addend
74     *
75     * @return The sum
76     */
77     function add(uint256 numA_, uint256 numB_) internal pure returns (uint256) {
78         uint256 numC_ = numA_ + numB_;
79         assert(numC_ >= numA_);
80         return numC_;
81     }
82 }
83
84 /**
85  *
86  * @title Genesis Contract
87  *
88  * Initializes events, modifiers & data in the contract and defines default functionality
89  * that follows the ERC20 standard token format
90  *
91  */
92 contract Genesis {
93     using SafeMath for uint256;                                // Use SafeMath library to test the logic of uint256 ca
94
95     // Initialise contract global constants
96     string constant ERR_PERMISSION_DENIED = "Permission denied!"; // Error message 101
97
98     // Initialise token information
99     string public name;                                         // Token Name
100    string public symbol;                                       // Token Symbol
101    uint8 public decimals;                                     // Token decimals (droplets)
102    address coinOwner ;                                         // Token owners address
103    uint256 coinSupply;                                         // Total token supply
104    mapping(address => uint256) balances;                       // Token balance state
105    mapping(address => mapping (address => uint256)) allowed;    // Token allowance state
106
107    // Owner privileges only
108    modifier ownerOnly() {
109        require(msg.sender == coinOwner, ERR_PERMISSION_DENIED) ;
110        _;
111    }
112
113    // Transfer tokens
114    event Transfer(address indexed owner_, address indexed receiver_, uint256 tokens_);
115
116    // Approve token allowances
117    event Approval(address indexed owner_, address indexed delegate_, uint256 tokens_);
118
119    // Fallback function handles unidentified calls and allows contract to receive payments
120    function() payable external { }
121
122    // @return total supply of tokens
123    function totalSupply() external view returns (uint256 supply) { return coinSupply; }
124
125    /** Gets the balance of tokens for an address
126    * @param owner_ The address owning The tokens
127    *
128    * @return The balance of tokens
129    */
130    function balanceOf(address owner_) external view returns (uint balance) { return balances[owner_]; }
131
132    /** Transfer tokens to an address
133    * @param receiver_ The address of the recipient
134    * @param tokens_ The amount of token to be transferred
135    *
136    * @return Whether the approval was successful or not
137    */
138    function transfer(address receiver_, uint tokens_) public returns (bool success) {}

```

```
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 success) {}
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 success) { }
164 }
```