CertiK Audit Report For Rupiah Token (rupiahtoken.com) - IDRT

Request Date: 2019-03-27
Revision Date: 2019-07-03

CERTIK

Contents

Disclaimer

This Report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and Rupiah Token (rupiahtoken.com) - IDRT(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 1.4B in assets.

For more information: https://certik.org/

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by Rupiah Token (rupiahtoken.com) - IDRT. This audit was conducted to discover issues and vulnerabilities in the source code of Rupiah Token (rupiahtoken.com) - IDRT's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

For every issues found, CertiK categorizes them into 3 buckets based on its risk level:

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.

Low

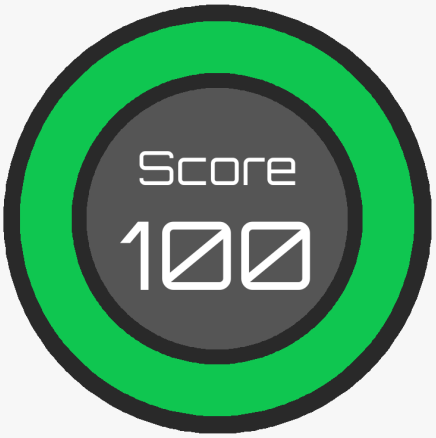
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

CERTIK *believes this smart contract passes security qualifications to be listed on digital asset exchanges.*

Jul 03, 2019



Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

"tx.origin" for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 51 in File ERC20RupiahTokenV1.sol

```
51 pragma solidity ^0.4.25;
```

Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 25 in File Pausable.sol

```
25 pragma solidity ^0.4.25;
```

Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 24 in File SafeMath.sol

```
24 pragma solidity ^0.4.25;
```

Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 24 in File Ownable.sol

```
24 pragma solidity ^0.4.25;
```

Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 60 in File Blacklistable.sol

```
60 pragma solidity ^0.4.25;
```

Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 60 in File IDRTWalletV1.sol

```
60 pragma solidity ^0.4.25;
```

Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 44 in File MultiSigWallet.sol

```
44 pragma solidity ^0.4.25;
```

Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

CERTIK *label location*: Line 30-34 in File howtoread.sol

CERTIK *label*

```
30  /*@CTK FAIL "transferFrom to same address"
31      @tag assume_completion
32      @pre from == to
33      @post __post.allowed[from][msg.sender] ==
34  */
```

Raw code location: Line 35-41 in File howtoread.sol

Raw code

```
35  function transferFrom(address from, address to
        ) {
36      balances[from] = balances[from].sub(tokens
37      allowed[from][msg.sender] = allowed[from][
38      balances[to] = balances[to].add(tokens);
39      emit Transfer(from, to, tokens);
40      return true;
41  }
```

Counterexample: This code violates the specification

Initial environment

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
53             balance: 0x0
54         }
55     }
56
```

Post environment

```
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

initialize

03, Jul 2019
68.79 ms

Line 81-87 in File ERC20RupiahTokenV1.sol

```
81    /*@CTK initialize
82      @post __post.owner == msg.sender
83      @post __post._name == name
84      @post __post._symbol == symbol
85      @post __post._currency == currency
86      @post __post._decimals == decimals
87     */
```

Line 88-94 in File ERC20RupiahTokenV1.sol

```
88    function initialize(string name, string symbol, string currency, uint8 decimals)
          initializer public {
89  owner = msg.sender;
90        _name = name;
91        _symbol = symbol;
92        _currency = currency;
93        _decimals = decimals;
94    }
```

The code meets the specification.

Formal Verification Request 2

name

03, Jul 2019
4.67 ms

Line 99-101 in File ERC20RupiahTokenV1.sol

```
99    /*@CTK name
100     @post __return == _name
101    */
```

Line 102-104 in File ERC20RupiahTokenV1.sol

```
102   function name() public view returns (string memory) {
103       return _name;
104   }
```

The code meets the specification.

Formal Verification Request 3

symbol

03, Jul 2019
4.58 ms

Line 109-111 in File ERC20RupiahTokenV1.sol

```
109     /*@CTK symbol
110       @post __return == _symbol
111      */
```

Line 112-114 in File ERC20RupiahTokenV1.sol

```
112     function symbol() public view returns (string memory) {
113         return _symbol;
114     }
```

The code meets the specification.

Formal Verification Request 4

currency

03, Jul 2019
4.43 ms

Line 119-121 in File ERC20RupiahTokenV1.sol

```
119     /*@CTK currency
120       @post __return == _currency
121      */
```

Line 122-124 in File ERC20RupiahTokenV1.sol

```
122     function currency() public view returns (string memory) {
123         return _currency;
124     }
```

The code meets the specification.

Formal Verification Request 5

decimals

03, Jul 2019
4.1 ms

Line 129-131 in File ERC20RupiahTokenV1.sol

```
129     /*@CTK decimals
130       @post __return == _decimals
131      */
```

Line 132-134 in File ERC20RupiahTokenV1.sol

```
132     function decimals() public view returns (uint8) {
133         return _decimals;
134     }
```

The code meets the specification.

Formal Verification Request 6

totalSupply

📅 03, Jul 2019
⏱ 4.5 ms

Line 139-141 in File ERC20RupiahTokenV1.sol

```
139     /*@CTK totalSupply
140       @post __return == _totalSupply
141      */
```

Line 142-144 in File ERC20RupiahTokenV1.sol

```
142     function totalSupply() public view returns (uint256) {
143         return _totalSupply;
144     }
```

✅ The code meets the specification.

Formal Verification Request 7

balanceOf

📅 03, Jul 2019
⏱ 5.13 ms

Line 151-153 in File ERC20RupiahTokenV1.sol

```
151     /*@CTK balanceOf
152       @post __return == _balances[owner]
153      */
```

Line 154-156 in File ERC20RupiahTokenV1.sol

```
154     function balanceOf(address owner) public view returns (uint256) {
155         return _balances[owner];
156     }
```

✅ The code meets the specification.

Formal Verification Request 8

allowance

📅 03, Jul 2019
⏱ 4.58 ms

Line 164-166 in File ERC20RupiahTokenV1.sol

```
164     /*@CTK allowance
165       @post __return == _allowed[owner][spender]
166      */
```

Line 167-169 in File ERC20RupiahTokenV1.sol

```
167     function allowance(address owner, address spender) public view returns (uint256) {
168         return _allowed[owner][spender];
169     }
```

The code meets the specification.

Formal Verification Request 9

transfer

03, Jul 2019
286.25 ms

Line 176-185 in File ERC20RupiahTokenV1.sol

```
176     /*@CTK transfer
177       @tag assume_completion
178       @pre msg.sender != to
179       @post _paused == false
180       @post blacklisted[msg.sender] == false
181       @post blacklisted[to] == false
182       @post to != address(0)
183       @post __post._balances[msg.sender] == _balances[msg.sender] - value
184       @post __post._balances[to] == _balances[to] + value
185      */
```

Line 186-194 in File ERC20RupiahTokenV1.sol

```
186     function transfer(address to, uint256 value) public whenNotPaused notBlacklisted(
            msg.sender) notBlacklisted(to) returns (bool) {
187         require(to != address(0));
188
189         _balances[msg.sender] = _balances[msg.sender].sub(value);
190         _balances[to] = _balances[to].add(value);
191         emit Transfer(msg.sender, to, value);
192
193         return true;
194     }
```

The code meets the specification.

Formal Verification Request 10

transferFrom

03, Jul 2019
518.88 ms

Line 218-228 in File ERC20RupiahTokenV1.sol

```
218     /*@CTK transferFrom
219       @tag assume_completion
220       @pre from != to
221       @post _paused == false
222       @post blacklisted[msg.sender] == false
223       @post blacklisted[from] == false
```

```
224      @post blacklisted[to] == false
225      @post to != address(0)
226      @post __post._balances[from] == _balances[from] - value
227      @post __post._balances[to] == _balances[to] + value
228     */
```

Line 229-238 in File ERC20RupiahTokenV1.sol

```
229    function transferFrom(address from, address to, uint256 value) public
           whenNotPaused notBlacklisted(msg.sender) notBlacklisted(from) notBlacklisted(
           to) returns (bool) {
230        require(to != address(0));
231
232        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
233
234        _balances[from] = _balances[from].sub(value);
235        _balances[to] = _balances[to].add(value);
236        emit Transfer(from, to, value);
237        return true;
238    }
```

The code meets the specification.

Formal Verification Request 11

increaseAllowance

03, Jul 2019
159.94 ms

Line 250-256 in File ERC20RupiahTokenV1.sol

```
250    /*@CTK increaseAllowance
251      @tag assume_completion
252      @post _paused == false
253      @post !blacklisted[msg.sender]
254      @post !blacklisted[spender]
255      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
             addedValue
256     */
```

Line 257-260 in File ERC20RupiahTokenV1.sol

```
257    function increaseAllowance(address spender, uint256 addedValue) public
           whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
           ) {
258        _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
259        return true;
260    }
```

The code meets the specification.

Formal Verification Request 12

decreaseAllowance

03, Jul 2019
136.66 ms

Line 272-278 in File ERC20RupiahTokenV1.sol

```
272     /*@CTK decreaseAllowance
273       @tag assume_completion
274       @post _paused == false
275       @post !blacklisted[msg.sender]
276       @post !blacklisted[spender]
277       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
              subtractedValue
278      */
```

Line 279-282 in File ERC20RupiahTokenV1.sol

```
279     function decreaseAllowance(address spender, uint256 subtractedValue) public
            whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
            ) {
280         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
                ));
281         return true;
282     }
```

The code meets the specification.

Formal Verification Request 13

_approve

03, Jul 2019
4.74 ms

Line 337-342 in File ERC20RupiahTokenV1.sol

```
337     /*@CTK _approve
338       @tag assume_completion
339       @post spender != address(0)
340       @post owner != address(0)
341       @post __post._allowed[owner][spender] == value
342      */
```

Line 343-349 in File ERC20RupiahTokenV1.sol

```
343     function _approve(address owner, address spender, uint256 value) internal {
344         require(spender != address(0));
345         require(owner != address(0));
346 
347         _allowed[owner][spender] = value;
348         emit Approval(owner, spender, value);
349     }
```

The code meets the specification.

Formal Verification Request 14

paused

03, Jul 2019
5.74 ms

Line 42-44 in File Pausable.sol

```
42    /*@CTK paused
43      @post __return == _paused
44     */
```

Line 45-47 in File Pausable.sol

```
45    function paused() public view returns (bool) {
46        return _paused;
47    }
```

The code meets the specification.

Formal Verification Request 15

pause

03, Jul 2019
12.28 ms

Line 68-72 in File Pausable.sol

```
68    /*@CTK pause
69      @tag assume_completion
70      @post owner == msg.sender
71      @post __post._paused == true
72     */
```

Line 73-76 in File Pausable.sol

```
73    function pause() public onlyOwner {
74        _paused = true;
75        emit Paused(msg.sender);
76    }
```

The code meets the specification.

Formal Verification Request 16

unpause

03, Jul 2019
13.61 ms

Line 81-85 in File Pausable.sol

```
81    /*@CTK unpause
82      @tag assume_completion
83      @post owner == msg.sender
84      @post __post._paused == false
85     */
```

Line 86-89 in File Pausable.sol

```
86    function unpause() public onlyOwner {
87        _paused = false;
88        emit Unpaused(msg.sender);
89    }
```

The code meets the specification.

Formal Verification Request 17

SafeMath_mul

03, Jul 2019
311.59 ms

Line 34-41 in File SafeMath.sol

```
34    /*@CTK SafeMath_mul
35      @post __reverted == __has_overflow
36      @post __reverted == false -> __return == a * b
37      @post a == 0 -> __return == 0
38      @post msg == msg__post
39      @post (a > 0 && (a * b / a != b)) == __reverted
40      @post __addr_map == __addr_map__post
41    */
```

Line 42-54 in File SafeMath.sol

```
42    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
43        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
44        // benefit is lost if 'b' is also tested.
45        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
46        if (a == 0) {
47            return 0;
48        }
49
50        uint256 c = a * b;
51        require(c / a == b);
52
53        return c;
54    }
```

The code meets the specification.

Formal Verification Request 18

SafeMath div

03, Jul 2019
11.64 ms

Line 59-63 in File SafeMath.sol

```
59    /*@CTK "SafeMath div"
60      @post b != 0 -> !__reverted
61      @post !__reverted -> __return == a / b
62      @post !__reverted -> !__has_overflow
63    */
```

Line 64-71 in File SafeMath.sol

```
64    function div(uint256 a, uint256 b) internal pure returns (uint256) {
65        // Solidity only automatically asserts when dividing by 0
66        require(b > 0);
67        uint256 c = a / b;
68        // assert(a == b * c + a % b); // There is no case in which this doesn't hold
69
70        return c;
71    }
```

The code meets the specification.

Formal Verification Request 19

SafeMath sub

03, Jul 2019
10.48 ms

Line 76-80 in File SafeMath.sol

```
76    /*@CTK "SafeMath sub"
77      @post (a < b) == __reverted
78      @post !__reverted -> __return == a - b
79      @post !__reverted -> !__has_overflow
80    */
```

Line 81-86 in File SafeMath.sol

```
81    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
82        require(b <= a);
83        uint256 c = a - b;
84
85        return c;
86    }
```

The code meets the specification.

Formal Verification Request 20

SafeMath_add

03, Jul 2019
12.84 ms

Line 91-97 in File SafeMath.sol

```
91    /*@CTK SafeMath_add
92      @post __reverted == __has_overflow
93      @post __reverted == false -> __return == a + b
94      @post msg == msg__post
95      @post (a + b < a) == __has_overflow
96      @post __addr_map == __addr_map__post
97     */
```

Line 98-103 in File SafeMath.sol

```
98    function add(uint256 a, uint256 b) internal pure returns (uint256) {
99        uint256 c = a + b;
100       require(c >= a);
101
102       return c;
103   }
```

The code meets the specification.

Formal Verification Request 21

SafeMath_mod

03, Jul 2019
10.74 ms

Line 109-112 in File SafeMath.sol

```
109   /*@CTK SafeMath_mod
110     @tag assume_completion
111     @post __return == a % b
112    */
```

Line 113-116 in File SafeMath.sol

```
113   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
114       require(b != 0);
115       return a % b;
116   }
```

The code meets the specification.

Formal Verification Request 22

Ownable

03, Jul 2019
4.83 ms

Line 43-45 in File Ownable.sol

```
43  /*@CTK Ownable
44    @post __post.owner == msg.sender
45   */
```

Line 46-48 in File Ownable.sol

```
46  constructor() public {
47    owner = msg.sender;
48  }
```

The code meets the specification.

Formal Verification Request 23

renounceOwnership

03, Jul 2019
13.9 ms

Line 64-68 in File Ownable.sol

```
64  /*@CTK renounceOwnership
65    @tag assume_completion
66    @post __post.owner == address(0)
67    @post owner == msg.sender
68   */
```

Line 69-72 in File Ownable.sol

```
69  function renounceOwnership() public onlyOwner {
70    owner = address(0);
71    emit OwnershipTransferred(msg.sender, owner);
72  }
```

The code meets the specification.

Formal Verification Request 24

transferOwnership

03, Jul 2019
42.94 ms

Line 78-83 in File Ownable.sol

```
78  /*@CTK transferOwnership
79    @tag assume_completion
80    @post owner == msg.sender
81    @post _newOwner != address(0)
82    @post __post.owner == _newOwner
83   */
```

Line 84-86 in File Ownable.sol

```
84  function transferOwnership(address _newOwner) public onlyOwner {
85    _transferOwnership(_newOwner);
86  }
```

The code meets the specification.

Formal Verification Request 25

_transferOwnership

03, Jul 2019
1.42 ms

Line 92-96 in File Ownable.sol

```
/*@CTK _transferOwnership
  @tag assume_completion
  @post _newOwner != address(0)
  @post __post.owner == _newOwner
 */
```

Line 97-101 in File Ownable.sol

```
function _transferOwnership(address _newOwner) internal {
  require(_newOwner != address(0));
  owner = _newOwner;
  emit OwnershipTransferred(owner, _newOwner);
}
```

✅ The code meets the specification.

Formal Verification Request 26

isBlacklisted

📅 03, Jul 2019
⏱ 4.68 ms

Line 87-89 in File Blacklistable.sol

```
  /*@CTK isBlacklisted
    @post __return == blacklisted[_account]
   */
```

Line 90-92 in File Blacklistable.sol

```
  function isBlacklisted(address _account) public view returns (bool) {
      return blacklisted[_account];
  }
```

✅ The code meets the specification.

Formal Verification Request 27

blacklist

📅 03, Jul 2019
⏱ 20.93 ms

Line 98-103 in File Blacklistable.sol

```
  /*@CTK blacklist
    @tag assume_completion
    @post owner == msg.sender
    @post _paused == false
    @post __post.blacklisted[_account]
   */
```

Line 104-107 in File Blacklistable.sol

```
  function blacklist(address _account) public onlyOwner whenNotPaused {
      blacklisted[_account] = true;
      emit Blacklisted(_account);
  }
```

The code meets the specification.

Formal Verification Request 28

unblacklist

03, Jul 2019
23.3 ms

Line 113-118 in File Blacklistable.sol

```
113     /*@CTK unblacklist
114       @tag assume_completion
115       @post owner == msg.sender
116       @post _paused == false
117       @post __post.blacklisted[_account] == false
118      */
```

Line 119-122 in File Blacklistable.sol

```
119     function unblacklist(address _account) public onlyOwner whenNotPaused {
120         blacklisted[_account] = false;
121         emit Unblacklisted(_account);
122     }
```

The code meets the specification.

Formal Verification Request 29

setPrintLimit

03, Jul 2019
14.42 ms

Line 200-204 in File IDRTWalletV1.sol

```
200     /*@CTK setPrintLimit
201       @tag assume_completion
202       @post msg.sender == _superOwner
203       @post __post._printLimit == newLimit
204      */
```

Line 205-211 in File IDRTWalletV1.sol

```
205     function setPrintLimit(uint256 newLimit)
206         public
207         onlySuperOwner()
208     {
209         emit PrintLimitChanged(_printLimit, newLimit);
210         _printLimit = newLimit;
211     }
```

The code meets the specification.

Formal Verification Request 30

transferOwnership

📅 03, Jul 2019
⏱ 21.11 ms

Line 217-221 in File IDRTWalletV1.sol

```
217     /*@CTK transferOwnership
218       @tag assume_completion
219       @post msg.sender == _superOwner
220       @post newAddress != address(0)
221      */
```

Line 222-230 in File IDRTWalletV1.sol

```
222     function transferOwnership(address newAddress)
223         public
224         onlySuperOwner()
225     {
226         require(newAddress != address(0));
227 
228         _superOwner = newAddress;
229         emit OwnershipTransferred(msg.sender, newAddress);
230     }
```

✅ The code meets the specification.

Formal Verification Request 31

superOwner

📅 03, Jul 2019
⏱ 4.84 ms

Line 235-237 in File IDRTWalletV1.sol

```
235     /*@CTK superOwner
236       @post __return == _superOwner
237      */
```

Line 238-243 in File IDRTWalletV1.sol

```
238     function superOwner()
239         public view
240         returns (address)
241     {
242         return _superOwner;
243     }
```

✅ The code meets the specification.

Formal Verification Request 32

requireFinalization

📅 03, Jul 2019
⏱ 4.57 ms

Line 249-251 in File IDRTWalletV1.sol

```
249     /*@CTK requireFinalization
250       @post __return == _requireFinalization[transactionId]
251      */
```

Line 252-257 in File IDRTWalletV1.sol

```
252     function requireFinalization(uint transactionId)
253         public view
254         returns (bool)
255     {
256         return _requireFinalization[transactionId];
257     }
```

The code meets the specification.

Formal Verification Request 33

addOwner

03, Jul 2019
56.72 ms

Line 167-174 in File MultiSigWallet.sol

```
167     /*@CTK addOwner
168       @tag assume_completion
169       @post msg.sender == address(this)
170       @post !isOwner[owner]
171       @post __post.isOwner[owner]
172       @post owner != 0
173       @post __post.owners[owners.length] == owner
174      */
```

Line 175-185 in File MultiSigWallet.sol

```
175     function addOwner(address owner)
176         public
177         onlyWallet
178         ownerDoesNotExist(owner)
179         notNull(owner)
180         validRequirement(owners.length + 1, required)
181     {
182         isOwner[owner] = true;
183         owners.push(owner);
184         emit OwnerAddition(owner);
185     }
```

The code meets the specification.

Formal Verification Request 34

changeRequirement

03, Jul 2019
26.33 ms

Line 236-241 in File MultiSigWallet.sol

```
236     /*@CTK changeRequirement
237       @tag assume_completion
238       @post msg.sender == address(this)
239       @post owners.length >= _required
240       @post __post.required == _required
241      */
```

Line 242-249 in File MultiSigWallet.sol

```
242     function changeRequirement(uint _required)
243         public
244         onlyWallet
245         validRequirement(owners.length, _required)
246     {
247         required = _required;
248         emit RequirementChange(_required);
249     }
```

The code meets the specification.

Formal Verification Request 35

revokeConfirmation

03, Jul 2019
42.91 ms

Line 286-292 in File MultiSigWallet.sol

```
286     /*@CTK revokeConfirmation
287       @tag assume_completion
288       @post isOwner[msg.sender]
289       @post confirmations[transactionId][msg.sender]
290       @post !transactions[transactionId].executed
291       @post __post.confirmations[transactionId][msg.sender] == false
292      */
```

Line 293-301 in File MultiSigWallet.sol

```
293     function revokeConfirmation(uint transactionId)
294         public
295         ownerExists(msg.sender)
296         confirmed(transactionId, msg.sender)
297         notExecuted(transactionId)
298     {
299         confirmations[transactionId][msg.sender] = false;
300         emit Revocation(msg.sender, transactionId);
301     }
```

The code meets the specification.

Formal Verification Request 36

getOwners

📅 03, Jul 2019
⏱ 4.65 ms

Line 437-439 in File MultiSigWallet.sol

```
437     /*@CTK getOwners
438       @post __return == owners
439      */
```

Line 440-446 in File MultiSigWallet.sol

```
440     function getOwners()
441         public
442         constant
443         returns (address[])
444     {
445         return owners;
446     }
```

✅ The code meets the specification.

Source Code with CertiK Labels

File token/ERC20RupiahTokenV1.sol

```
1  /**
2   * Rupiah Token Smart Contract
3   * Copyright (C) 2019 PT. Rupiah Token Indonesia <https://www.rupiahtoken.com/>.
4   *
5   * This program is free software: you can redistribute it and/or modify
6   * it under the terms of the GNU Affero General Public License as published by
7   * the Free Software Foundation, either version 3 of the License, or
8   * (at your option) any later version.
9   *
10  * This program is distributed in the hope that it will be useful,
11  * but WITHOUT ANY WARRANTY; without even the implied warranty of
12  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13  * GNU Affero General Public License for more details.
14  *
15  * You should have received a copy of the GNU Affero General Public License
16  * along with this program. If not, see <http://www.gnu.org/licenses/>.
17  *
18  * This file incorporates work covered byt the following copyright and
19  * permission notice:
20  *
21  *     OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
22  *     Copyright (c) 2016 Smart Contract Solutions, Inc.
23  *     Modified for Rupiah Token by FengkieJ 2019.
24  *
25  *     centre-tokens <https://github.com/centrehq/centre-tokens>
26  *     Copyright CENTRE SECZ 2018.
27  *     Modified for Rupiah Token by FengkieJ 2019.
28  *
29  *     ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
30  *     Copyright (c) 2018 ZeppelinOS Global Limited.
31  *
32  *     The MIT License (MIT)
33  *
34  *     Permission is hereby granted, free of charge, to any person obtaining a copy
35  *     of this software and associated documentation files (the "Software"), to deal
36  *     in the Software without restriction, including without limitation the rights
37  *     to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
38  *     copies of the Software, and to permit persons to whom the Software is furnished
        to
39  *     do so, subject to the following conditions:
40  *
41  *     The above copyright notice and this permission notice shall be included in all
42  *     copies or substantial portions of the Software.
43  *
44  *     THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
45  *     IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
46  *     FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
47  *     AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
      LIABILITY,
48  *     WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
49  *     CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
50  */
51 pragma solidity ^0.4.25;
52
```

```
53  import "./IERC20.sol";
54  import "../math/SafeMath.sol";
55  import "../governance/Blacklistable.sol";
56  import "../zos/Initializable.sol";
57
58  /**
59   * @title ERC20RupiahToken
60   * @dev ERC20 compliant fiat token that is backed by Indonesian Rupiah 1:1
61   */
62  contract ERC20RupiahToken is IERC20, Blacklistable, Initializable {
63      using SafeMath for uint256;
64
65      string internal _name;
66      string internal _symbol;
67      string internal _currency;
68      uint8 internal _decimals;
69
70      mapping (address => uint256) internal _balances;
71      mapping (address => mapping (address => uint256)) internal _allowed;
72      uint256 internal _totalSupply;
73
74      /**
75       * @dev Initialize the smart contract to work with ZeppelinOS, can only be called
             once.
76       * @param name describes the name of the token.
77       * @param symbol describes the symbol of the token.
78       * @param currency describes the currency of the token.
79       * @param decimals describes the number of decimals of the token.
80       */
81      /*@CTK initialize
82        @post __post.owner == msg.sender
83        @post __post._name == name
84        @post __post._symbol == symbol
85        @post __post._currency == currency
86        @post __post._decimals == decimals
87       */
88      function initialize(string name, string symbol, string currency, uint8 decimals)
            initializer public {
89    owner = msg.sender;
90          _name = name;
91          _symbol = symbol;
92          _currency = currency;
93          _decimals = decimals;
94      }
95
96      /**
97       * @return the name of the token.
98       */
99      /*@CTK name
100       @post __return == _name
101      */
102     function name() public view returns (string memory) {
103         return _name;
104     }
105
106     /**
107      * @return the symbol of the token.
108      */
```

```
109     /*@CTK symbol
110       @post __return == _symbol
111      */
112     function symbol() public view returns (string memory) {
113         return _symbol;
114     }
115 
116     /**
117      * @return the currency of the token.
118      */
119     /*@CTK currency
120       @post __return == _currency
121      */
122     function currency() public view returns (string memory) {
123         return _currency;
124     }
125 
126     /**
127      * @return the number of decimals of the token.
128      */
129     /*@CTK decimals
130       @post __return == _decimals
131      */
132     function decimals() public view returns (uint8) {
133         return _decimals;
134     }
135 
136     /**
137      * @return the total number of tokens in existence
138      */
139     /*@CTK totalSupply
140       @post __return == _totalSupply
141      */
142     function totalSupply() public view returns (uint256) {
143         return _totalSupply;
144     }
145 
146     /**
147     * @dev Gets the balance of the specified address.
148     * @param owner The address to query the balance of.
149     * @return An uint256 representing the amount owned by the passed address.
150     */
151     /*@CTK balanceOf
152       @post __return == _balances[owner]
153      */
154     function balanceOf(address owner) public view returns (uint256) {
155         return _balances[owner];
156     }
157 
158     /**
159      * @dev Function to check the amount of tokens that an owner allowed to a spender.
160      * @param owner address The address which owns the funds.
161      * @param spender address The address which will spend the funds.
162      * @return A uint256 specifying the amount of tokens still available for the
           spender.
163      */
164     /*@CTK allowance
165       @post __return == _allowed[owner][spender]
```

```
166      */
167     function allowance(address owner, address spender) public view returns (uint256) {
168         return _allowed[owner][spender];
169     }
170
171     /**
172     * @dev Transfer token for a specified address
173     * @param to The address to transfer to.
174     * @param value The amount to be transferred.
175     */
176     /*@CTK transfer
177       @tag assume_completion
178       @pre msg.sender != to
179       @post _paused == false
180       @post blacklisted[msg.sender] == false
181       @post blacklisted[to] == false
182       @post to != address(0)
183       @post __post._balances[msg.sender] == _balances[msg.sender] - value
184       @post __post._balances[to] == _balances[to] + value
185      */
186     function transfer(address to, uint256 value) public whenNotPaused notBlacklisted(
          msg.sender) notBlacklisted(to) returns (bool) {
187         require(to != address(0));
188
189         _balances[msg.sender] = _balances[msg.sender].sub(value);
190         _balances[to] = _balances[to].add(value);
191         emit Transfer(msg.sender, to, value);
192
193         return true;
194     }
195
196     /**
197      * @dev Approve the passed address to spend the specified amount of tokens on
           behalf of msg.sender.
198      * Beware that changing an allowance with this method brings the risk that someone
           may use both the old
199      * and the new allowance by unfortunate transaction ordering. One possible
           solution to mitigate this
200      * race condition is to first reduce the spender's allowance to 0 and set the
           desired value afterwards:
201      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
202      * @param spender The address which will spend the funds.
203      * @param value The amount of tokens to be spent.
204      */
205     function approve(address spender, uint256 value) public whenNotPaused
          notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool) {
206         _approve(msg.sender, spender, value);
207         return true;
208     }
209
210     /**
211      * @dev Transfer tokens from one address to another.
212      * Note that while this function emits an Approval event, this is not required as
           per the specification,
213      * and other compliant implementations may not emit the event.
214      * @param from address The address which you want to send tokens from
215      * @param to address The address which you want to transfer to
216      * @param value uint256 the amount of tokens to be transferred
```

```
217      */
218     /*@CTK transferFrom
219       @tag assume_completion
220       @pre from != to
221       @post _paused == false
222       @post blacklisted[msg.sender] == false
223       @post blacklisted[from] == false
224       @post blacklisted[to] == false
225       @post to != address(0)
226       @post __post._balances[from] == _balances[from] - value
227       @post __post._balances[to] == _balances[to] + value
228      */
229     function transferFrom(address from, address to, uint256 value) public
          whenNotPaused notBlacklisted(msg.sender) notBlacklisted(from) notBlacklisted(
          to) returns (bool) {
230         require(to != address(0));
231
232         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
233
234         _balances[from] = _balances[from].sub(value);
235         _balances[to] = _balances[to].add(value);
236         emit Transfer(from, to, value);
237         return true;
238     }
239
240     /**
241      * @dev Increase the amount of tokens that an owner allowed to a spender.
242      * approve should be called when allowed_[_spender] == 0. To increment
243      * allowed value is better to use this function to avoid 2 calls (and wait until
244      * the first transaction is mined)
245      * From MonolithDAO Token.sol
246      * Emits an Approval event.
247      * @param spender The address which will spend the funds.
248      * @param addedValue The amount of tokens to increase the allowance by.
249      */
250     /*@CTK increaseAllowance
251       @tag assume_completion
252       @post _paused == false
253       @post !blacklisted[msg.sender]
254       @post !blacklisted[spender]
255       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
              addedValue
256      */
257     function increaseAllowance(address spender, uint256 addedValue) public
          whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
          ) {
258         _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
259         return true;
260     }
261
262     /**
263      * @dev Decrease the amount of tokens that an owner allowed to a spender.
264      * approve should be called when allowed_[_spender] == 0. To decrement
265      * allowed value is better to use this function to avoid 2 calls (and wait until
266      * the first transaction is mined)
267      * From MonolithDAO Token.sol
268      * Emits an Approval event.
269      * @param spender The address which will spend the funds.
```

```
270      * @param subtractedValue The amount of tokens to decrease the allowance by.
271      */
272     /*@CTK decreaseAllowance
273       @tag assume_completion
274       @post _paused == false
275       @post !blacklisted[msg.sender]
276       @post !blacklisted[spender]
277       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
              subtractedValue
278      */
279     function decreaseAllowance(address spender, uint256 subtractedValue) public
             whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
             ) {
280         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
                ));
281         return true;
282     }
283
284     /**
285      * @dev Function that mints an amount of the token and assigns it to
286      * an account. This encapsulates the modification of balances such that the
287      * proper events are emitted.
288      * @param account The account that will receive the created tokens.
289      * @param value The amount that will be created.
290      */
291     function mint(address account, uint256 value) public whenNotPaused notBlacklisted(
             account) onlyOwner {
292         require(account != address(0));
293
294         value = value.mul(10**_decimals);
295         _totalSupply = _totalSupply.add(value);
296         _balances[account] = _balances[account].add(value);
297         emit Transfer(address(0), account, value);
298     }
299
300     /**
301      * @dev Function that burns an amount of the token.
302      * @param value The amount that will be burnt.
303      */
304     function burn(uint256 value) public whenNotPaused onlyOwner {
305         value = value.mul(10**_decimals);
306
307         _totalSupply = _totalSupply.sub(value);
308         _balances[msg.sender] = _balances[msg.sender].sub(value);
309         emit Transfer(msg.sender, address(0), value);
310     }
311
312     /**
313      * @dev Function that burns an amount of the token of a given
314      * account, deducting from the sender's allowance for said account. Uses the
315      * internal burn function.
316      * Emits an Approval event (reflecting the reduced allowance).
317      * @param account The account whose tokens will be burnt.
318      * @param value The amount that will be burnt.
319      */
320     function burnFrom(address account, uint256 value) public whenNotPaused
             notBlacklisted(account) onlyOwner {
321         require(account != address(0));
```

```
322
323         value = value.mul(10**_decimals);
324         _totalSupply = _totalSupply.sub(value);
325         _balances[account] = _balances[account].sub(value);
326         emit Transfer(account, address(0), value);
327
328         _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
329     }
330
331     /**
332      * @dev Approve an address to spend another addresses' tokens.
333      * @param owner The address that owns the tokens.
334      * @param spender The address that will spend the tokens.
335      * @param value The number of tokens that can be spent.
336      */
337     /*@CTK _approve
338       @tag assume_completion
339       @post spender != address(0)
340       @post owner != address(0)
341       @post __post._allowed[owner][spender] == value
342      */
343     function _approve(address owner, address spender, uint256 value) internal {
344         require(spender != address(0));
345         require(owner != address(0));
346
347         _allowed[owner][spender] = value;
348         emit Approval(owner, spender, value);
349     }
350 }
```

File lifecycle/Pausable.sol

```
1  /**
2   * The MIT License (MIT)
3   *
4   * OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
5   * Copyright (c) 2016 Smart Contract Solutions, Inc.
6   * Modified for Rupiah Token by FengkieJ 2019.
7   *
8   * Permission is hereby granted, free of charge, to any person obtaining a copy
9   * of this software and associated documentation files (the "Software"), to deal
10  * in the Software without restriction, including without limitation the rights
11  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
12  * copies of the Software, and to permit persons to whom the Software is furnished to
13  * do so, subject to the following conditions:
14  *
15  * The above copyright notice and this permission notice shall be included in all
16  * copies or substantial portions of the Software.
17  *
18  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
19  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
20  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
21  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
22  * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
23  * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
24  */
25 pragma solidity ^0.4.25;
26
27 import "../ownership/Ownable.sol";
```

```
28
29  /**
30   * @title Pausable
31   * @dev Base contract which allows children to implement an emergency stop mechanism.
32   */
33  contract Pausable is Ownable {
34      event Paused(address account);
35      event Unpaused(address account);
36
37      bool private _paused;
38
39      /**
40       * @return true if the contract is paused, false otherwise.
41       */
42      /*@CTK paused
43        @post __return == _paused
44       */
45      function paused() public view returns (bool) {
46          return _paused;
47      }
48
49      /**
50       * @dev Modifier to make a function callable only when the contract is not paused.
51       */
52      modifier whenNotPaused() {
53          require(!_paused);
54          _;
55      }
56
57      /**
58       * @dev Modifier to make a function callable only when the contract is paused.
59       */
60      modifier whenPaused() {
61          require(_paused);
62          _;
63      }
64
65      /**
66       * @dev called by the owner to pause, triggers stopped state
67       */
68      /*@CTK pause
69        @tag assume_completion
70        @post owner == msg.sender
71        @post __post._paused == true
72       */
73      function pause() public onlyOwner {
74          _paused = true;
75          emit Paused(msg.sender);
76      }
77
78      /**
79       * @dev called by the owner to unpause, returns to normal state
80       */
81      /*@CTK unpause
82        @tag assume_completion
83        @post owner == msg.sender
84        @post __post._paused == false
85       */
```

```
86      function unpause() public onlyOwner {
87          _paused = false;
88          emit Unpaused(msg.sender);
89      }
90  }
```

File math/SafeMath.sol

```
1   /**
2    * The MIT License (MIT)
3    *
4    * OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
5    * Copyright (c) 2016 Smart Contract Solutions, Inc.
6    *
7    * Permission is hereby granted, free of charge, to any person obtaining a copy
8    * of this software and associated documentation files (the "Software"), to deal
9    * in the Software without restriction, including without limitation the rights
10   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11   * copies of the Software, and to permit persons to whom the Software is furnished to
12   * do so, subject to the following conditions:
13   *
14   * The above copyright notice and this permission notice shall be included in all
15   * copies or substantial portions of the Software.
16   *
17   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
21   * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
22   * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
23   */
24  pragma solidity ^0.4.25;
25
26  /**
27   * @title SafeMath
28   * @dev Unsigned math operations with safety checks that revert on error
29   */
30  library SafeMath {
31      /**
32      * @dev Multiplies two unsigned integers, reverts on overflow.
33      */
34      /*@CTK SafeMath_mul
35        @post __reverted == __has_overflow
36        @post __reverted == false -> __return == a * b
37        @post a == 0 -> __return == 0
38        @post msg == msg__post
39        @post (a > 0 && (a * b / a != b)) == __reverted
40        @post __addr_map == __addr_map__post
41      */
42      function mul(uint256 a, uint256 b) internal pure returns (uint256) {
43          // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
44          // benefit is lost if 'b' is also tested.
45          // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
46          if (a == 0) {
47              return 0;
48          }
49
50          uint256 c = a * b;
51          require(c / a == b);
```

```
52
53          return c;
54      }
55
56      /**
57      * @dev Integer division of two unsigned integers truncating the quotient, reverts
            on division by zero.
58      */
59      /*@CTK "SafeMath div"
60        @post b != 0 -> !__reverted
61        @post !__reverted -> __return == a / b
62        @post !__reverted -> !__has_overflow
63      */
64      function div(uint256 a, uint256 b) internal pure returns (uint256) {
65          // Solidity only automatically asserts when dividing by 0
66          require(b > 0);
67          uint256 c = a / b;
68          // assert(a == b * c + a % b); // There is no case in which this doesn't hold
69
70          return c;
71      }
72
73      /**
74      * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is
            greater than minuend).
75      */
76      /*@CTK "SafeMath sub"
77        @post (a < b) == __reverted
78        @post !__reverted -> __return == a - b
79        @post !__reverted -> !__has_overflow
80      */
81      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
82          require(b <= a);
83          uint256 c = a - b;
84
85          return c;
86      }
87
88      /**
89      * @dev Adds two unsigned integers, reverts on overflow.
90      */
91      /*@CTK SafeMath_add
92        @post __reverted == __has_overflow
93        @post __reverted == false -> __return == a + b
94        @post msg == msg__post
95        @post (a + b < a) == __has_overflow
96        @post __addr_map == __addr_map__post
97       */
98      function add(uint256 a, uint256 b) internal pure returns (uint256) {
99          uint256 c = a + b;
100         require(c >= a);
101
102         return c;
103     }
104
105     /**
106     * @dev Divides two unsigned integers and returns the remainder (unsigned integer
            modulo),
```

```
    * reverts when dividing by zero.
    */
    /*@CTK SafeMath_mod
      @tag assume_completion
      @post __return == a % b
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}
```

File ownership/Ownable.sol

```
/**
 * The MIT License (MIT)
 *
 * OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
 * Copyright (c) 2016 Smart Contract Solutions, Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is furnished to
 * do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
pragma solidity ^0.4.25;

/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization
     control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
  address public owner;

  event OwnershipTransferred(
    address indexed previousOwner,
    address indexed newOwner
  );

  /**
   * @dev The Ownable constructor sets the original `owner` of the contract to the
       sender
   * account.
   */
  /*@CTK Ownable
```

```
44    @post __post.owner == msg.sender
45   */
46  constructor() public {
47    owner = msg.sender;
48  }
49
50  /**
51   * @dev Throws if called by any account other than the owner.
52   */
53  modifier onlyOwner() {
54    require(msg.sender == owner);
55    _;
56  }
57
58  /**
59   * @dev Allows the current owner to relinquish control of the contract.
60   * @notice Renouncing to ownership will leave the contract without an owner.
61   * It will not be possible to call the functions with the `onlyOwner`
62   * modifier anymore.
63   */
64  /*@CTK renounceOwnership
65    @tag assume_completion
66    @post __post.owner == address(0)
67    @post owner == msg.sender
68   */
69  function renounceOwnership() public onlyOwner {
70    owner = address(0);
71    emit OwnershipTransferred(msg.sender, owner);
72  }
73
74  /**
75   * @dev Allows the current owner to transfer control of the contract to a newOwner.
76   * @param _newOwner The address to transfer ownership to.
77   */
78  /*@CTK transferOwnership
79    @tag assume_completion
80    @post owner == msg.sender
81    @post _newOwner != address(0)
82    @post __post.owner == _newOwner
83   */
84  function transferOwnership(address _newOwner) public onlyOwner {
85    _transferOwnership(_newOwner);
86  }
87
88  /**
89   * @dev Transfers control of the contract to a newOwner.
90   * @param _newOwner The address to transfer ownership to.
91   */
92  /*@CTK _transferOwnership
93    @tag assume_completion
94    @post _newOwner != address(0)
95    @post __post.owner == _newOwner
96   */
97  function _transferOwnership(address _newOwner) internal {
98    require(_newOwner != address(0));
99    owner = _newOwner;
100   emit OwnershipTransferred(owner, _newOwner);
101 }
```

```
102 }
```

File governance/Blacklistable.sol

```
1  /**
2   * Rupiah Token Smart Contract
3   * Copyright (C) 2019 PT. Rupiah Token Indonesia <https://www.rupiahtoken.com/>.
4   *
5   * This program is free software: you can redistribute it and/or modify
6   * it under the terms of the GNU Affero General Public License as published by
7   * the Free Software Foundation, either version 3 of the License, or
8   * (at your option) any later version.
9   *
10  * This program is distributed in the hope that it will be useful,
11  * but WITHOUT ANY WARRANTY; without even the implied warranty of
12  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13  * GNU Affero General Public License for more details.
14  *
15  * You should have received a copy of the GNU Affero General Public License
16  * along with this program. If not, see <http://www.gnu.org/licenses/>.
17  *
18  * This file incorporates work covered by the following copyright and
19  * permission notice:
20  *
21  *     Ethereum Multisignature Wallet <https://github.com/gnosis/MultiSigWallet>
22  *     Copyright (c) 2016 Gnosis Ltd.
23  *     Modified for Rupiah Token by FengkieJ 2019.
24  *
25  *     This program is free software: you can redistribute it and/or modify
26  *     it under the terms of the GNU Lesser General Public License as published by
27  *     the Free Software Foundation, either version 3 of the License, or
28  *     (at your option) any later version.
29  *
30  *     This program is distributed in the hope that it will be useful,
31  *     but WITHOUT ANY WARRANTY; without even the implied warranty of
32  *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
33  *     GNU Lesser General Public License for more details.
34  *
35  *     You should have received a copy of the GNU Lesser General Public License
36  *     along with this program. If not, see <http://www.gnu.org/licenses/>.
37  *-------------------------------------------------------------------------------------------

38  *     ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
39  *     Copyright (c) 2018 ZeppelinOS Global Limited.
40  *
41  *     The MIT License (MIT)
42  *
43  *     Permission is hereby granted, free of charge, to any person obtaining a copy
44  *     of this software and associated documentation files (the "Software"), to deal
45  *     in the Software without restriction, including without limitation the rights
46  *     to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
47  *     copies of the Software, and to permit persons to whom the Software is furnished
        to
48  *     do so, subject to the following conditions:
49  *
50  *     The above copyright notice and this permission notice shall be included in all
51  *     copies or substantial portions of the Software.
52  *
53  *     THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
```

```
54  *     IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
55  *     FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
56  *     AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
        LIABILITY,
57  *     WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
58  *     CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
59  */
60  pragma solidity ^0.4.25;
61
62  import "../lifecycle/Pausable.sol";
63
64  /**
65   * @title Blacklistable
66   * @dev Allows accounts to be blacklisted by a "blacklister" role
67  */
68  contract Blacklistable is Pausable {
69      mapping(address => bool) internal blacklisted;
70
71      event Blacklisted(address indexed _account);
72      event Unblacklisted(address indexed _account);
73
74      /**
75       * @dev Throws if argument account is blacklisted
76       * @param _account The address to check
77      */
78      modifier notBlacklisted(address _account) {
79          require(blacklisted[_account] == false);
80          _;
81      }
82
83      /**
84       * @dev Checks if account is blacklisted
85       * @param _account The address to check
86      */
87      /*@CTK isBlacklisted
88        @post __return == blacklisted[_account]
89       */
90      function isBlacklisted(address _account) public view returns (bool) {
91          return blacklisted[_account];
92      }
93
94      /**
95       * @dev Adds account to blacklist
96       * @param _account The address to blacklist
97      */
98      /*@CTK blacklist
99        @tag assume_completion
100       @post owner == msg.sender
101       @post _paused == false
102       @post __post.blacklisted[_account]
103      */
104     function blacklist(address _account) public onlyOwner whenNotPaused {
105         blacklisted[_account] = true;
106         emit Blacklisted(_account);
107     }
108
109     /**
110      * @dev Removes account from blacklist
```

```
111      * @param _account The address to remove from the blacklist
112     */
113     /*@CTK unblacklist
114       @tag assume_completion
115       @post owner == msg.sender
116       @post _paused == false
117       @post __post.blacklisted[_account] == false
118      */
119     function unblacklist(address _account) public onlyOwner whenNotPaused {
120         blacklisted[_account] = false;
121         emit Unblacklisted(_account);
122     }
123 }
```

File governance/wallet/IDRTWalletV1.sol

```
1  /**
2   * Rupiah Token Smart Contract
3   * Copyright (C) 2019 PT. Rupiah Token Indonesia <https://www.rupiahtoken.com/>.
4   *
5   * This program is free software: you can redistribute it and/or modify
6   * it under the terms of the GNU Affero General Public License as published by
7   * the Free Software Foundation, either version 3 of the License, or
8   * (at your option) any later version.
9   *
10  * This program is distributed in the hope that it will be useful,
11  * but WITHOUT ANY WARRANTY; without even the implied warranty of
12  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13  * GNU Affero General Public License for more details.
14  *
15  * You should have received a copy of the GNU Affero General Public License
16  * along with this program. If not, see <http://www.gnu.org/licenses/>.
17  *
18  * This file incorporates work covered by the following copyright and
19  * permission notice:
20  *
21  *     Ethereum Multisignature Wallet <https://github.com/gnosis/MultiSigWallet>
22  *     Copyright (c) 2016 Gnosis Ltd.
23  *     Modified for Rupiah Token by FengkieJ 2019.
24  *
25  *     This program is free software: you can redistribute it and/or modify
26  *     it under the terms of the GNU Lesser General Public License as published by
27  *     the Free Software Foundation, either version 3 of the License, or
28  *     (at your option) any later version.
29  *
30  *     This program is distributed in the hope that it will be useful,
31  *     but WITHOUT ANY WARRANTY; without even the implied warranty of
32  *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
33  *     GNU Lesser General Public License for more details.
34  *
35  *     You should have received a copy of the GNU Lesser General Public License
36  *     along with this program. If not, see <http://www.gnu.org/licenses/>.
37  *-------------------------------------------------------------------------------

38  *     ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
39  *     Copyright (c) 2018 ZeppelinOS Global Limited.
40  *
41  *     The MIT License (MIT)
42  *
```

```
43  *     Permission is hereby granted, free of charge, to any person obtaining a copy
44  *     of this software and associated documentation files (the "Software"), to deal
45  *     in the Software without restriction, including without limitation the rights
46  *     to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
47  *     copies of the Software, and to permit persons to whom the Software is furnished
        to
48  *     do so, subject to the following conditions:
49  *
50  *     The above copyright notice and this permission notice shall be included in all
51  *     copies or substantial portions of the Software.
52  *
53  *     THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
54  *     IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
55  *     FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
56  *     AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
       LIABILITY,
57  *     WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
58  *     CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
59  */
60  pragma solidity ^0.4.25;
61
62  import "./MultiSigWallet.sol";
63
64  contract IDRTWallet is MultiSigWallet {
65      uint256 internal _printLimit;
66      mapping (uint => bool) internal _requireFinalization;
67      address internal _superOwner;
68
69      event OwnershipTransferred(
70          address indexed previousOwner,
71          address indexed newOwner
72      );
73
74      event PrintLimitChanged(
75          uint256 indexed oldValue,
76          uint256 indexed newValue
77      );
78
79      event RequireFinalization(uint indexed transactionId);
80
81      event Finalized(uint indexed transactionId);
82
83      /**
84       * @dev Throws if called by any account other than _superOwner.
85       */
86      modifier onlySuperOwner() {
87          require(msg.sender == _superOwner);
88          _;
89      }
90
91      /**
92       * @dev Initialize the smart contract to work with ZeppelinOS, can only be called
            once.
93       * @param admins list of the multisig contract admins.
94       * @param required number of required confirmations to execute a transaction.
95       * @param printLimit maximum amount of minting limit before _superOwner need to
            finalize.
96       */
```

```
97      function initialize(address[] admins, uint256 required, uint256 printLimit) public
            initializer {
98          MultiSigWallet.initialize(admins, required);
99          _superOwner = msg.sender;
100         _printLimit = printLimit;
101     }
102
103     /**
104      * @dev Get the function signature from call data.
105      * @param data the call data in bytes.
106      * @return function signature in bytes4.
107      */
108     function getFunctionSignature(bytes memory data) internal pure returns (bytes4 out
            ) {
109         assembly {
110             out := mload(add(data, 0x20))
111         }
112     }
113
114     /**
115      * @dev Get the value to mint from call data.
116      * @param data the call data in bytes.
117      * @return value to mint in uint256.
118      */
119     function getValueToMint(bytes memory data) internal pure returns (uint256 value) {
120         bytes32 x;
121         assembly {
122             x := mload(add(data, 0x44))
123         }
124         value = uint256(x);
125     }
126
127     /**
128      * @dev Allows an owner to submit and confirm a transaction.
129      * @param destination Transaction target address.
130      * @param value Transaction ether value.
131      * @param data Transaction data payload.
132      * @return the transaction ID.
133      */
134     function submitTransaction(address destination, uint value, bytes data)
135         public
136         returns (uint transactionId)
137     {
138         transactionId = addTransaction(destination, value, data);
139         bytes4 functionSignature = getFunctionSignature(data);
140     if(
141             (functionSignature == 0x99a88ec4) || //ZeppelinOS ProxyAdmin.sol's upgrade
                    function
142             (functionSignature == 0x9623609d) || //ZeppelinOS ProxyAdmin.sol's
                    upgradeAndCall function
143             (functionSignature == 0xe20056e6) || //MultiSigWallet.sol's replaceOwner
                    function
144             (functionSignature == 0x7065cb48) || //MultiSigWallet.sol's addOwner
                    function
145             (functionSignature == 0x173825d9) || //MultiSigWallet.sol's removeOwner
                    function
146             (functionSignature == 0x715018a6) || //ERC20 Ownable's renounceOwnership
                    function
```

```
147             (functionSignature == 0xf2fde38b) || //ERC20 Ownable's transferOwnership
                    function
148             ((functionSignature == 0x40c10f19) && (getValueToMint(data) > _printLimit))
                     //Calls mint function and value exceeds _printLimit
149         ) {
150             _requireFinalization[transactionId] = true;
151             emit RequireFinalization(transactionId);
152         }
153         confirmTransaction(transactionId);
154     }
155
156     /**
157      * @dev Allows anyone to execute a confirmed transaction.
158      * @param transactionId Transaction ID.
159      */
160     function executeTransaction(uint transactionId)
161         public
162         ownerExists(msg.sender)
163         confirmed(transactionId, msg.sender)
164         notExecuted(transactionId)
165     {
166         if(!_requireFinalization[transactionId]) {
167             super.executeTransaction(transactionId);
168         } else {
169             emit RequireFinalization(transactionId);
170         }
171     }
172
173     /**
174      * @dev Finalize tx by _superOwner.
175      * @param transactionId Transaction ID.
176      */
177     function finalizeTransaction(uint transactionId)
178         public
179         onlySuperOwner()
180         notExecuted(transactionId)
181     {
182         require(_requireFinalization[transactionId]);
183         require(isConfirmed(transactionId));
184
185         Transaction storage txn = transactions[transactionId];
186         txn.executed = true;
187         if (external_call(txn.destination, txn.value, txn.data.length, txn.data)) {
188             emit Execution(transactionId);
189             emit Finalized(transactionId);
190         } else {
191             emit ExecutionFailure(transactionId);
192             txn.executed = false;
193         }
194     }
195
196     /**
197      * @dev Set new printLimit before _superOwner need to finalize.
198      * @param newLimit of print limit amount.
199      */
200     /*@CTK setPrintLimit
201       @tag assume_completion
202       @post msg.sender == _superOwner
```

```
203       @post __post._printLimit == newLimit
204     */
205     function setPrintLimit(uint256 newLimit)
206         public
207         onlySuperOwner()
208     {
209         emit PrintLimitChanged(_printLimit, newLimit);
210         _printLimit = newLimit;
211     }
212
213     /**
214      * @dev Set new _superOwner address.
215      * @param newAddress new address for _superOwner
216      */
217     /*@CTK transferOwnership
218       @tag assume_completion
219       @post msg.sender == _superOwner
220       @post newAddress != address(0)
221      */
222     function transferOwnership(address newAddress)
223         public
224         onlySuperOwner()
225     {
226         require(newAddress != address(0));
227
228         _superOwner = newAddress;
229         emit OwnershipTransferred(msg.sender, newAddress);
230     }
231
232     /**
233      * @dev Get current _superOwner address.
234      */
235     /*@CTK superOwner
236       @post __return == _superOwner
237      */
238     function superOwner()
239         public view
240         returns (address)
241     {
242         return _superOwner;
243     }
244
245
246     /**
247      * @dev Get whether a transaction require finalization or not.
248      */
249     /*@CTK requireFinalization
250       @post __return == _requireFinalization[transactionId]
251      */
252     function requireFinalization(uint transactionId)
253         public view
254         returns (bool)
255     {
256         return _requireFinalization[transactionId];
257     }
258 }
```

File governance/wallet/MultiSigWallet.sol

```
/**
 * Ethereum Multisignature Wallet <https://github.com/gnosis/MultiSigWallet>
 * Copyright (c) 2016 Gnosis Ltd.
 * Modified for Rupiah Token by FengkieJ 2019.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * 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. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * This file incorporates work covered by the following copyright and
 * permission notice:
 *     ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
 *     Copyright (c) 2018 ZeppelinOS Global Limited.
 *
 *     The MIT License (MIT)
 *
 *     Permission is hereby granted, free of charge, to any person obtaining a copy
 *     of this software and associated documentation files (the "Software"), to deal
 *     in the Software without restriction, including without limitation the rights
 *     to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 *     copies of the Software, and to permit persons to whom the Software is furnished
      to
 *     do so, subject to the following conditions:
 *
 *     The above copyright notice and this permission notice shall be included in all
 *     copies or substantial portions of the Software.
 *
 *     THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 *     IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 *     FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 *     AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
     LIABILITY,
 *     WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 *     CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

pragma solidity ^0.4.25;

import "../../zos/Initializable.sol";

/// @title Multisignature wallet - Allows multiple parties to agree on transactions
    before execution.
/// @author Stefan George - <stefan.george@consensys.net>
/// Modified for Rupiah Token by FengkieJ 2019

contract MultiSigWallet is Initializable {
    /*
     * Events
     */
```

```
56      event Confirmation(address indexed sender, uint indexed transactionId);
57      event Revocation(address indexed sender, uint indexed transactionId);
58      event Submission(uint indexed transactionId);
59      event Execution(uint indexed transactionId);
60      event ExecutionFailure(uint indexed transactionId);
61      event Deposit(address indexed sender, uint value);
62      event OwnerAddition(address indexed owner);
63      event OwnerRemoval(address indexed owner);
64      event RequirementChange(uint required);
65
66      /*
67       *  Constants
68       */
69      uint constant public MAX_OWNER_COUNT = 50;
70
71      /*
72       *  Storage
73       */
74      mapping (uint => Transaction) public transactions;
75      mapping (uint => mapping (address => bool)) public confirmations;
76      mapping (address => bool) public isOwner;
77      address[] public owners;
78      uint public required;
79      uint public transactionCount;
80
81      struct Transaction {
82          address destination;
83          uint value;
84          bytes data;
85          bool executed;
86      }
87
88      /*
89       *  Modifiers
90       */
91      modifier onlyWallet() {
92          require(msg.sender == address(this));
93          _;
94      }
95
96      modifier ownerDoesNotExist(address owner) {
97          require(!isOwner[owner]);
98          _;
99      }
100
101     modifier ownerExists(address owner) {
102         require(isOwner[owner]);
103         _;
104     }
105
106     modifier transactionExists(uint transactionId) {
107         require(transactions[transactionId].destination != 0);
108         _;
109     }
110
111     modifier confirmed(uint transactionId, address owner) {
112         require(confirmations[transactionId][owner]);
113         _;
```

```
114     }
115
116     modifier notConfirmed(uint transactionId, address owner) {
117         require(!confirmations[transactionId][owner]);
118         _;
119     }
120
121     modifier notExecuted(uint transactionId) {
122         require(!transactions[transactionId].executed);
123         _;
124     }
125
126     modifier notNull(address _address) {
127         require(_address != 0);
128         _;
129     }
130
131     modifier validRequirement(uint ownerCount, uint _required) {
132         require(ownerCount <= MAX_OWNER_COUNT
133             && _required <= ownerCount
134             && _required != 0
135             && ownerCount != 0);
136         _;
137     }
138
139     /// @dev Fallback function allows to deposit ether.
140     function()
141         payable
142     {
143         if (msg.value > 0)
144             emit Deposit(msg.sender, msg.value);
145     }
146
147     /*
148      * Public functions
149      */
150     /// @dev Initializer sets initial owners and required number of confirmations.
151     /// @param _owners List of initial owners.
152     /// @param _required Number of required confirmations.
153     function initialize(address[] _owners, uint _required)
154         public
155         validRequirement(_owners.length, _required) initializer
156     {
157   for (uint i=0; i<_owners.length; i++) {
158             require(!isOwner[_owners[i]] && _owners[i] != 0);
159             isOwner[_owners[i]] = true;
160         }
161         owners = _owners;
162         required = _required;
163     }
164
165     /// @dev Allows to add a new owner. Transaction has to be sent by wallet.
166     /// @param owner Address of new owner.
167     /*@CTK addOwner
168       @tag assume_completion
169       @post msg.sender == address(this)
170       @post !isOwner[owner]
171       @post __post.isOwner[owner]
```

```
172       @post owner != 0
173       @post __post.owners[owners.length] == owner
174      */
175     function addOwner(address owner)
176         public
177         onlyWallet
178         ownerDoesNotExist(owner)
179         notNull(owner)
180         validRequirement(owners.length + 1, required)
181     {
182         isOwner[owner] = true;
183         owners.push(owner);
184         emit OwnerAddition(owner);
185     }
186
187     /// @dev Allows to remove an owner. Transaction has to be sent by wallet.
188     /// @param owner Address of owner.
189     function removeOwner(address owner)
190         public
191         onlyWallet
192         ownerExists(owner)
193     {
194         isOwner[owner] = false;
195         /*CTK find_owner_index
196           @inv this == this__pre
197           @inv owners == owners__pre
198           @inv i <= owners.length - 1
199           @inv !__should_return
200           @inv i > 0 -> owners[i - 1] != owner
201           @post i == owners.length - 1 || owners[i] == owner
202          */
203         for (uint i=0; i<owners.length - 1; i++)
204             if (owners[i] == owner) {
205                 owners[i] = owners[owners.length - 1];
206                 break;
207             }
208         owners.length -= 1;
209         if (required > owners.length)
210             changeRequirement(owners.length);
211         emit OwnerRemoval(owner);
212     }
213
214     /// @dev Allows to replace an owner with a new owner. Transaction has to be sent
        by wallet.
215     /// @param owner Address of owner to be replaced.
216     /// @param newOwner Address of new owner.
217     function replaceOwner(address owner, address newOwner)
218         public
219         onlyWallet
220         ownerExists(owner)
221         ownerDoesNotExist(newOwner)
222     {
223         for (uint i=0; i<owners.length; i++)
224             if (owners[i] == owner) {
225                 owners[i] = newOwner;
226                 break;
227             }
228         isOwner[owner] = false;
```

```
229         isOwner[newOwner] = true;
230         emit OwnerRemoval(owner);
231         emit OwnerAddition(newOwner);
232     }
233 
234     /// @dev Allows to change the number of required confirmations. Transaction has to
          be sent by wallet.
235     /// @param _required Number of required confirmations.
236     /*@CTK changeRequirement
237       @tag assume_completion
238       @post msg.sender == address(this)
239       @post owners.length >= _required
240       @post __post.required == _required
241      */
242     function changeRequirement(uint _required)
243         public
244         onlyWallet
245         validRequirement(owners.length, _required)
246     {
247         required = _required;
248         emit RequirementChange(_required);
249     }
250 
251     /// @dev Allows an owner to submit and confirm a transaction.
252     /// @param destination Transaction target address.
253     /// @param value Transaction ether value.
254     /// @param data Transaction data payload.
255     /// @return Returns transaction ID.
256     function submitTransaction(address destination, uint value, bytes data)
257         public
258         returns (uint transactionId)
259     {
260         transactionId = addTransaction(destination, value, data);
261         confirmTransaction(transactionId);
262     }
263 
264     /// @dev Allows an owner to confirm a transaction.
265     /// @param transactionId Transaction ID.
266     /*CTK confirmTransaction
267       @tag assume_completion
268       @post isOwner[msg.sender]
269       @post !confirmations[transactionId][msg.sender]
270       @post transactions[transactionId].destination != 0
271       @post __post.confirmations[transactionId][msg.sender]
272      */
273     function confirmTransaction(uint transactionId)
274         public
275         ownerExists(msg.sender)
276         transactionExists(transactionId)
277         notConfirmed(transactionId, msg.sender)
278     {
279         confirmations[transactionId][msg.sender] = true;
280         emit Confirmation(msg.sender, transactionId);
281         executeTransaction(transactionId);
282     }
283 
284     /// @dev Allows an owner to revoke a confirmation for a transaction.
285     /// @param transactionId Transaction ID.
```

```
286     /*@CTK revokeConfirmation
287       @tag assume_completion
288       @post isOwner[msg.sender]
289       @post confirmations[transactionId][msg.sender]
290       @post !transactions[transactionId].executed
291       @post __post.confirmations[transactionId][msg.sender] == false
292      */
293     function revokeConfirmation(uint transactionId)
294         public
295         ownerExists(msg.sender)
296         confirmed(transactionId, msg.sender)
297         notExecuted(transactionId)
298     {
299         confirmations[transactionId][msg.sender] = false;
300         emit Revocation(msg.sender, transactionId);
301     }
302
303     /// @dev Allows anyone to execute a confirmed transaction.
304     /// @param transactionId Transaction ID.
305     function executeTransaction(uint transactionId)
306         public
307         ownerExists(msg.sender)
308         confirmed(transactionId, msg.sender)
309         notExecuted(transactionId)
310     {
311         if (isConfirmed(transactionId)) {
312             Transaction storage txn = transactions[transactionId];
313             txn.executed = true;
314             if (external_call(txn.destination, txn.value, txn.data.length, txn.data))
315                 emit Execution(transactionId);
316             else {
317                 emit ExecutionFailure(transactionId);
318                 txn.executed = false;
319             }
320         }
321     }
322
323     // call has been separated into its own function in order to take advantage
324     // of the Solidity's code generator to produce a loop that copies tx.data into
            memory.
325     function external_call(address destination, uint value, uint dataLength, bytes
            data) internal returns (bool) {
326         bool result;
327         assembly {
328             let x := mload(0x40) // "Allocate" memory for output (0x40 is where "free
                memory" pointer is stored by convention)
329             let d := add(data, 32) // First 32 bytes are the padded length of data, so
                exclude that
330             result := call(
331                 sub(gas, 34710), // 34710 is the value that solidity is currently
                    emitting
332                                  // It includes callGas (700) + callVeryLow (3, to pay
                                        for SUB) + callValueTransferGas (9000) +
333                                  // callNewAccountGas (25000, in case the destination
                                        address does not exist and needs creating)
334                 destination,
335                 value,
336                 d,
```

```
337                 dataLength,      // Size of the input (in bytes) - this is what fixes
                        the padding problem
338                 x,
339                 0                // Output is ignored, therefore the output size is
                        zero
340             )
341         }
342         return result;
343     }
344 
345     /// @dev Returns the confirmation status of a transaction.
346     /// @param transactionId Transaction ID.
347     /// @return Confirmation status.
348     function isConfirmed(uint transactionId)
349         public
350         constant
351         returns (bool)
352     {
353         uint count = 0;
354         for (uint i=0; i<owners.length; i++) {
355             if (confirmations[transactionId][owners[i]])
356                 count += 1;
357             if (count == required)
358                 return true;
359         }
360     }
361 
362     /*
363      * Internal functions
364      */
365     /// @dev Adds a new transaction to the transaction mapping, if transaction does
            not exist yet.
366     /// @param destination Transaction target address.
367     /// @param value Transaction ether value.
368     /// @param data Transaction data payload.
369     /// @return Returns transaction ID.
370     /*CTK addTransaction
371       @tag assume_completion
372       @post destination != address(0)
373       @post __post.transactions[transactionCount].destination == destination
374       @post __post.transactions[transactionCount].value == value
375       @post __post.transactions[transactionCount].data == data
376       @post __post.transactions[transactionCount].executed == false
377       @post __post.transactionCount == transactionCount + 1
378      */
379     function addTransaction(address destination, uint value, bytes data)
380         internal
381         notNull(destination)
382         returns (uint transactionId)
383     {
384         transactionId = transactionCount;
385         transactions[transactionId].destination = destination;
386         transactions[transactionId].value = value;
387         transactions[transactionId].data = data;
388         transactions[transactionId].executed = false;
389         // transactions[transactionId] = Transaction({
390         //     destination: destination,
391         //     value: value,
```

```
392         //     data: data,
393         //     executed: false
394         // });
395         transactionCount += 1;
396         emit Submission(transactionId);
397     }
398 
399     /*
400      * Web3 call functions
401      */
402     /// @dev Returns number of confirmations of a transaction.
403     /// @param transactionId Transaction ID.
404     /// @return Number of confirmations.
405     function getConfirmationCount(uint transactionId)
406         public
407         constant
408         returns (uint count)
409     {
410         for (uint i=0; i<owners.length; i++)
411             if (confirmations[transactionId][owners[i]])
412                 count += 1;
413     }
414 
415     /// @dev Returns total number of transactions after filers are applied.
416     /// @param pending Include pending transactions.
417     /// @param executed Include executed transactions.
418     /// @return Total number of transactions after filters are applied.
419     //*CTK only two states are allowed. It is more readable to separate this
420     // function into multiple small ones. For example,
421     // 1. getTotalTransaction = 2 + 3
422     // 2. getPendingTransaction
423     // 3. getExecutedTransaction
424     function getTransactionCount(bool pending, bool executed)
425         public
426         constant
427         returns (uint count)
428     {
429         for (uint i=0; i<transactionCount; i++)
430             if (  pending && !transactions[i].executed
431                 || executed && transactions[i].executed)
432                 count += 1;
433     }
434 
435     /// @dev Returns list of owners.
436     /// @return List of owner addresses.
437     /*@CTK getOwners
438       @post __return == owners
439      */
440     function getOwners()
441         public
442         constant
443         returns (address[])
444     {
445         return owners;
446     }
447 
448     /// @dev Returns array with owner addresses, which confirmed transaction.
449     /// @param transactionId Transaction ID.
```

```
450     /// @return Returns array of owner addresses.
451     function getConfirmations(uint transactionId)
452         public
453         constant
454         returns (address[] _confirmations)
455     {
456         address[] memory confirmationsTemp = new address[](owners.length);
457         uint count = 0;
458         uint i;
459         for (i=0; i<owners.length; i++)
460             if (confirmations[transactionId][owners[i]]) {
461                 confirmationsTemp[count] = owners[i];
462                 count += 1;
463             }
464         _confirmations = new address[](count);
465         for (i=0; i<count; i++)
466             _confirmations[i] = confirmationsTemp[i];
467     }
468 
469     /// @dev Returns list of transaction IDs in defined range.
470     /// @param from Index start position of transaction array.
471     /// @param to Index end position of transaction array.
472     /// @param pending Include pending transactions.
473     /// @param executed Include executed transactions.
474     /// @return Returns array of transaction IDs.
475     //*CTK no checks for to > from.
476     // can be used by getTransactionCount
477     function getTransactionIds(uint from, uint to, bool pending, bool executed)
478         public
479         constant
480         returns (uint[] _transactionIds)
481     {
482         uint[] memory transactionIdsTemp = new uint[](transactionCount);
483         uint count = 0;
484         uint i;
485         for (i=0; i<transactionCount; i++)
486             if (  pending && !transactions[i].executed
487                 || executed && transactions[i].executed)
488             {
489                 transactionIdsTemp[count] = i;
490                 count += 1;
491             }
492         _transactionIds = new uint[](to - from);
493         for (i=from; i<to; i++)
494             _transactionIds[i - from] = transactionIdsTemp[i];
495     }
496 }
```