# CertiK Audit Report For Dapp

Request Date: 2019-06-07
Revision Date: 2019-08-06
Platform Name: Ethereum

# Contents

**Disclaimer** 1

**About CertiK** 2

**Exective Summary** 3

**Vulnerability Classification** 3

**Testing Summary** 4
- Audit Score . . . 4
- Type of Issues . . . 4
- Vulnerability Details . . . 5

**Manual Review Notes** 6

**Static Analysis Results** 7

**Formal Verification Results** 8
- How to read . . . 8

**Source Code with CertiK Labels** 26

## Disclaimer

## 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 6.2B in assets.

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

## Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by Dapp. This audit was conducted to discover issues and vulnerabilities in the source code of Dapp'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 issue 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 conditions, or it could affect the security standard by lost of access control.

**Low**

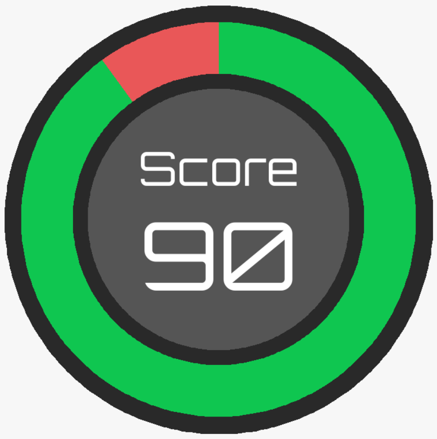
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

## Testing Summary

**WARNING**

CERTIK *identified some potential security flaws in this contract and also provided corresponding solutions.*

*Aug 06, 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. | 2 | 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. | 2 | 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. | 1 | 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

### Critical

No issue found.

### Medium

No issue found.

### Low

DappToken:

- `formatDecimals()`, `allocateToken()`: Potential numerical overflow. See *Review Notes* section for more details.

StandardToken:

- `transfer()`, `transferFrom()`: Potential numerical overflow. See *Review Notes* section for more details.
- `transfer()`, `transferFrom()`: Missing address check in the following methods, which may lead to value loss.

# Manual Review Notes

## Source Code SHA-256 Checksum

- **DAPPT.sol**
  `c30d8f162df79e643d597751c535b0021c5639707efcba5411bbeda0beb9736e`

## Disucssions

### DappToken

- MINOR `formatDecimals()`: Potential numerical overflow.
- MINOR `allocateToken()`: Potential numerical overflow but will be guarded by `totalSupply`.
- INFO `increaseSupply()`, `decreaseSupply()`: Potential numerical overflow to be captured by `safeAdd()` and `safeSubtract()`. However inconsistent event and resulting balance may happen in case of mistaken operation of the owner.

### StandardToken

- MINOR `transfer()`, `transferFrom()`: Potential numerical overflow but will be guarded by `totalSupply`.
- MINOR `transfer()`, `transferFrom()`: Missing address check which may lead to value loss:
- INFO Recommend using `require()` for condition check in transfer and transferFrom. The use of require() permits error message to be emitted for better diagnostic of transaction failure. Example:

```
require(to != address(0), ...)
require(from != to, ...)
require(balances[from] >= value, ...)
require(allowed[from][msg.sender] >= value, ...)
require(value > 0, ...)
```

- INFO Recommend changing the `if (...)throw` checks to `require(..., ...)`.
- MINOR Recommend using the pull model instead of the push model to better secure the ownership transfer.

```
address ethFundDeposit;
address proposedEthFundDeposit;
function proposeNewOwner(address newFundDeposit) isOwner() external {
  require(newFundDeposit != address(0), ...);
  proposedEthFundDeposit = newFundDeposit;
}
function claimNewOwner() external {
  require(msg.sender == proposedEthFundDeposit, ...);
  ethFundDeposit = proposedEthFundDeposit;
  proposedEthFundDeposit = address(0);
}
```

## Static Analysis Results

### INSECURE_COMPILER_VERSION

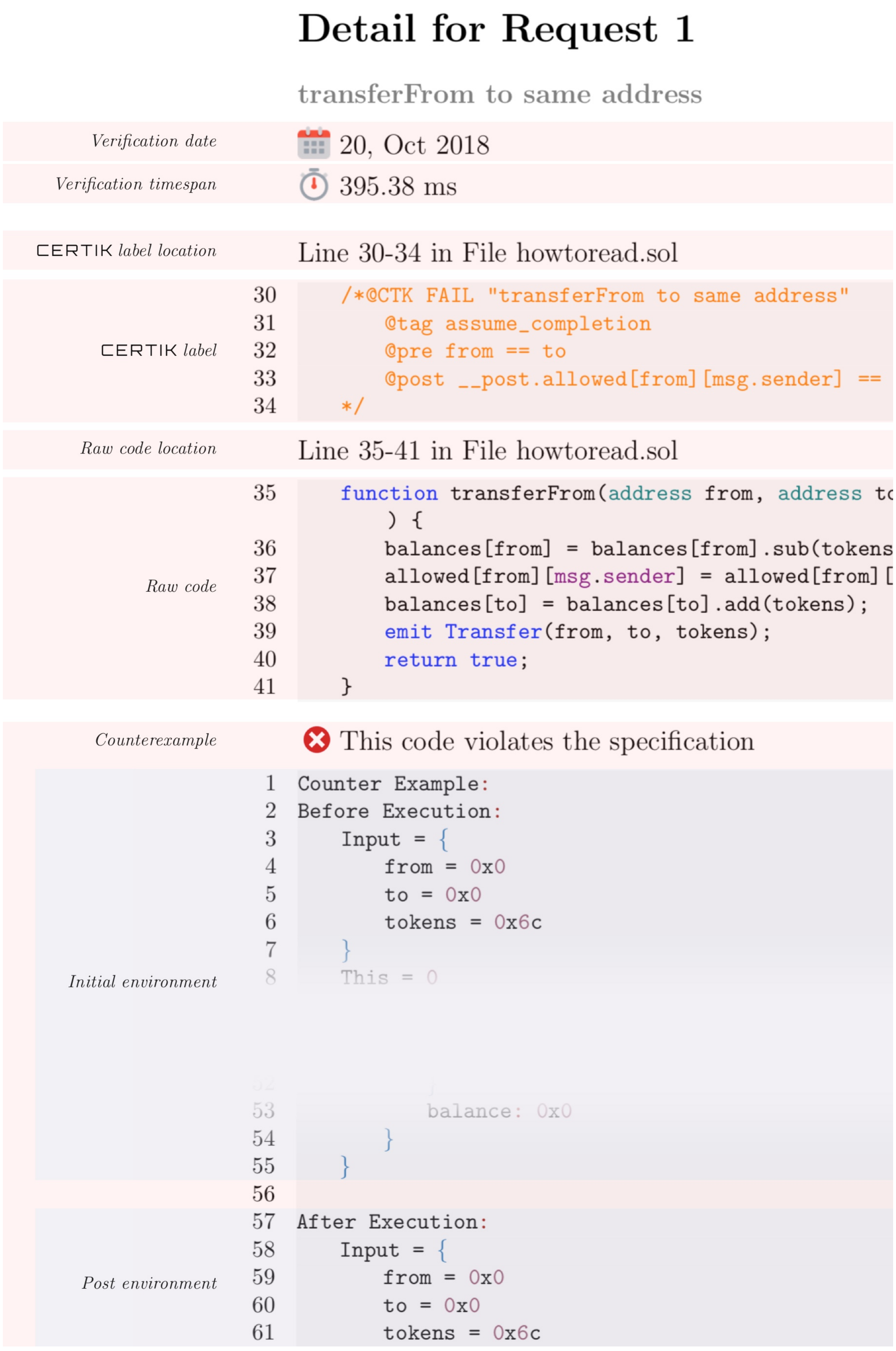
Line 5 in File DAPPT.sol

```
5 pragma solidity ^0.4.12;
```

❗ Version to compile has the following bug: 0.4.12: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput 0.4.13: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput 0.4.14: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue 0.4.15: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.16: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.17: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.18: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.19: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorag
ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.20: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 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

**Method will not encounter an assertion failure.**

06, Aug 2019
19.1 ms

Line 20 in File DAPPT.sol

```
    //@CTK FAIL NO_ASF
```

Line 28-32 in File DAPPT.sol

```
    function safeAdd(uint256 x, uint256 y) internal returns(uint256) {
      uint256 z = x + y;
      assert((z >= x) && (z >= y));
      return z;
    }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        x = 137
        y = 135
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "SafeMath",
          "balance": 0,
          "contract": {}
        }
      }
    ]

Function invocation is reverted.
```

## Formal Verification Request 2

### SafeMath add

06, Aug 2019
3.44 ms

Line 21-27 in File DAPPT.sol

```
    /*@CTK "SafeMath add"
      @post (x + y < x || x + y < y) == __reverted
      @post !__reverted -> __return == x + y
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
     */
```

Line 28-32 in File DAPPT.sol

```
    function safeAdd(uint256 x, uint256 y) internal returns(uint256) {
      uint256 z = x + y;
      assert((z >= x) && (z >= y));
      return z;
    }
```

✓ The code meets the specification.

## Formal Verification Request 3

### Method will not encounter an assertion failure.

06, Aug 2019
16.85 ms

Line 34 in File DAPPT.sol

```
    //@CTK FAIL NO_ASF
```

Line 42-46 in File DAPPT.sol

```
    function safeSubtract(uint256 x, uint256 y) internal returns(uint256) {
      assert(x >= y);
      uint256 z = x - y;
      return z;
    }
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        x = 0
        y = 1
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "SafeMath",
          "balance": 0,
          "contract": {}
        }
      }
    ]

Function invocation is reverted.
```

## Formal Verification Request 4

### SafeMath sub

📅 06, Aug 2019
⏱ 2.02 ms

Line 35-41 in File DAPPT.sol

```
    /*@CTK "SafeMath sub"
      @post (x < y) == __reverted
      @post !__reverted -> __return == x - y
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
     */
```

Line 42-46 in File DAPPT.sol

```
    function safeSubtract(uint256 x, uint256 y) internal returns(uint256) {
      assert(x >= y);
      uint256 z = x - y;
      return z;
    }
```

✅ The code meets the specification.

## Formal Verification Request 5

### Method will not encounter an assertion failure.

📅 06, Aug 2019
⏱ 27.76 ms

Line 48 in File DAPPT.sol

```
    //@CTK FAIL NO_ASF
```

Line 56-60 in File DAPPT.sol

```
    function safeMult(uint256 x, uint256 y) internal returns(uint256) {
      uint256 z = x * y;
      assert((x == 0)||(z/x == y));
      return z;
    }
```

✖ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        x = 11
        y = 159
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "SafeMath",
          "balance": 0,
          "contract": {}
        }
      }
    ]

Function invocation is reverted.
```

## Formal Verification Request 6

**If method completes, integer overflow would not happen.**

06, Aug 2019
5.81 ms

Line 130 in File DAPPT.sol

```
130     //@CTK NO_OVERFLOW
```

Line 137-139 in File DAPPT.sol

```
137     function balanceOf(address _owner) constant returns (uint256 balance) {
138         return balances[_owner];
139     }
```

✓ The code meets the specification.

## Formal Verification Request 7

**Buffer overflow / array index out of bound would never happen.**

06, Aug 2019
0.37 ms

Line 131 in File DAPPT.sol

```
131     //@CTK NO_BUF_OVERFLOW
```

Line 137-139 in File DAPPT.sol

```
137     function balanceOf(address _owner) constant returns (uint256 balance) {
138         return balances[_owner];
139     }
```

✓ The code meets the specification.

## Formal Verification Request 8

**Method will not encounter an assertion failure.**

06, Aug 2019
0.36 ms

Line 132 in File DAPPT.sol

```
132     //@CTK NO_ASF
```

Line 137-139 in File DAPPT.sol

```
137     function balanceOf(address _owner) constant returns (uint256 balance) {
138         return balances[_owner];
139     }
```

✓ The code meets the specification.

## Formal Verification Request 9

### balanceOf

📅 06, Aug 2019
⏱ 0.39 ms

Line 133-136 in File DAPPT.sol

```
/*@CTK balanceOf
  @tag assume_completion
  @post balance == (balances[_owner])
 */
```

Line 137-139 in File DAPPT.sol

```
function balanceOf(address _owner) constant returns (uint256 balance) {
    return balances[_owner];
}
```

✓ The code meets the specification.

## Formal Verification Request 10

### If method completes, integer overflow would not happen.

📅 06, Aug 2019
⏱ 10.26 ms

Line 141 in File DAPPT.sol

```
//@CTK NO_OVERFLOW
```

Line 149-153 in File DAPPT.sol

```
function approve(address _spender, uint256 _value) returns (bool success) {
    allowed[msg.sender][_spender] = _value;
    Approval(msg.sender, _spender, _value);
    return true;
}
```

✓ The code meets the specification.

## Formal Verification Request 11

### Buffer overflow / array index out of bound would never happen.

📅 06, Aug 2019
⏱ 0.4 ms

Line 142 in File DAPPT.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 149-153 in File DAPPT.sol

```
    function approve(address _spender, uint256 _value) returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }
```

✓ The code meets the specification.

## Formal Verification Request 12

### Method will not encounter an assertion failure.

06, Aug 2019
0.41 ms

Line 143 in File DAPPT.sol

```
    //@CTK NO_ASF
```

Line 149-153 in File DAPPT.sol

```
    function approve(address _spender, uint256 _value) returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }
```

✓ The code meets the specification.

## Formal Verification Request 13

### approve

06, Aug 2019
1.53 ms

Line 144-148 in File DAPPT.sol

```
    /*@CTK approve
      @tag assume_completion
      @post (__post.allowed[msg.sender][_spender]) == (_value)
      @post (success) == (true)
     */
```

Line 149-153 in File DAPPT.sol

```
    function approve(address _spender, uint256 _value) returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }
```

✓ The code meets the specification.

## Formal Verification Request 14

**If method completes, integer overflow would not happen.**

06, Aug 2019
6.3 ms

Line 155 in File DAPPT.sol

```
//@CTK NO_OVERFLOW
```

Line 163-165 in File DAPPT.sol

```
function allowance(address _owner, address _spender) constant returns (uint256
    remaining) {
  return allowed[_owner][_spender];
}
```

✓ The code meets the specification.

## Formal Verification Request 15

**Buffer overflow / array index out of bound would never happen.**

06, Aug 2019
0.43 ms

Line 156 in File DAPPT.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 163-165 in File DAPPT.sol

```
function allowance(address _owner, address _spender) constant returns (uint256
    remaining) {
  return allowed[_owner][_spender];
}
```

✓ The code meets the specification.

## Formal Verification Request 16

**Method will not encounter an assertion failure.**

06, Aug 2019
0.48 ms

Line 157 in File DAPPT.sol

```
//@CTK NO_ASF
```

Line 163-165 in File DAPPT.sol

```
function allowance(address _owner, address _spender) constant returns (uint256
    remaining) {
  return allowed[_owner][_spender];
}
```

✓ The code meets the specification.

## Formal Verification Request 17

allowance

📅 06, Aug 2019
⏱ 0.41 ms

Line 158-162 in File DAPPT.sol

```
/*@CTK allowance
  @tag assume_completion
  @post (__reverted) == (false)
  @post (remaining) == (__post.allowed[_owner][_spender])
 */
```

Line 163-165 in File DAPPT.sol

```
function allowance(address _owner, address _spender) constant returns (uint256
    remaining) {
  return allowed[_owner][_spender];
}
```

✅ The code meets the specification.

## Formal Verification Request 18

If method completes, integer overflow would not happen.

📅 06, Aug 2019
⏱ 27.81 ms

Line 242 in File DAPPT.sol

```
//@CTK NO_OVERFLOW
```

Line 252-257 in File DAPPT.sol

```
function setTokenExchangeRate(uint256 _tokenExchangeRate) isOwner external {
    if (_tokenExchangeRate == 0) throw;
    if (_tokenExchangeRate == tokenExchangeRate) throw;

    tokenExchangeRate = _tokenExchangeRate;
}
```

✅ The code meets the specification.

## Formal Verification Request 19

Buffer overflow / array index out of bound would never happen.

📅 06, Aug 2019
⏱ 0.53 ms

Line 243 in File DAPPT.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 252-257 in File DAPPT.sol

```
    function setTokenExchangeRate(uint256 _tokenExchangeRate) isOwner external {
        if (_tokenExchangeRate == 0) throw;
        if (_tokenExchangeRate == tokenExchangeRate) throw;

        tokenExchangeRate = _tokenExchangeRate;
    }
```

✓ The code meets the specification.

## Formal Verification Request 20

### Method will not encounter an assertion failure.

06, Aug 2019
0.52 ms

Line 244 in File DAPPT.sol

```
    //@CTK NO_ASF
```

Line 252-257 in File DAPPT.sol

```
    function setTokenExchangeRate(uint256 _tokenExchangeRate) isOwner external {
        if (_tokenExchangeRate == 0) throw;
        if (_tokenExchangeRate == tokenExchangeRate) throw;

        tokenExchangeRate = _tokenExchangeRate;
    }
```

✓ The code meets the specification.

## Formal Verification Request 21

### setTokenExchangeRate

06, Aug 2019
4.66 ms

Line 245-251 in File DAPPT.sol

```
    /*@CTK setTokenExchangeRate
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post _tokenExchangeRate != 0
      @post _tokenExchangeRate != tokenExchangeRate
      @post __post.tokenExchangeRate == _tokenExchangeRate
     */
```

Line 252-257 in File DAPPT.sol

```
    function setTokenExchangeRate(uint256 _tokenExchangeRate) isOwner external {
        if (_tokenExchangeRate == 0) throw;
        if (_tokenExchangeRate == tokenExchangeRate) throw;

        tokenExchangeRate = _tokenExchangeRate;
    }
```

✓ The code meets the specification.

## Formal Verification Request 22

**If method completes, integer overflow would not happen.**

📅 06, Aug 2019
⏱ 38.3 ms

Line 295 in File DAPPT.sol

```
    //@CTK NO_OVERFLOW
```

Line 308-316 in File DAPPT.sol

```
    function startFunding (uint256 _fundingStartBlock, uint256 _fundingStopBlock)
        isOwner external {
        if (isFunding) throw;
        if (_fundingStartBlock >= _fundingStopBlock) throw;
        if (block.number >= _fundingStartBlock) throw;

        fundingStartBlock = _fundingStartBlock;
        fundingStopBlock = _fundingStopBlock;
        isFunding = true;
    }
```

✅ The code meets the specification.

## Formal Verification Request 23

**Buffer overflow / array index out of bound would never happen.**

📅 06, Aug 2019
⏱ 0.69 ms

Line 296 in File DAPPT.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 308-316 in File DAPPT.sol

```
    function startFunding (uint256 _fundingStartBlock, uint256 _fundingStopBlock)
        isOwner external {
        if (isFunding) throw;
        if (_fundingStartBlock >= _fundingStopBlock) throw;
        if (block.number >= _fundingStartBlock) throw;

        fundingStartBlock = _fundingStartBlock;
        fundingStopBlock = _fundingStopBlock;
        isFunding = true;
    }
```

✅ The code meets the specification.

## Formal Verification Request 24

**Method will not encounter an assertion failure.**

📅 06, Aug 2019
⏱ 0.63 ms

Line 297 in File DAPPT.sol

```
    //@CTK NO_ASF
```

Line 308-316 in File DAPPT.sol

```
    function startFunding (uint256 _fundingStartBlock, uint256 _fundingStopBlock)
        isOwner external {
        if (isFunding) throw;
        if (_fundingStartBlock >= _fundingStopBlock) throw;
        if (block.number >= _fundingStartBlock) throw;

        fundingStartBlock = _fundingStartBlock;
        fundingStopBlock = _fundingStopBlock;
        isFunding = true;
    }
```

✓ The code meets the specification.

## Formal Verification Request 25

startFunding

06, Aug 2019
23.44 ms

Line 298-307 in File DAPPT.sol

```
    /*@CTK startFunding
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post !isFunding
      @post _fundingStartBlock < _fundingStopBlock
      @post block.number < _fundingStartBlock
      @post __post.fundingStartBlock == _fundingStartBlock
      @post __post.fundingStopBlock == _fundingStopBlock
      @post __post.isFunding
     */
```

Line 308-316 in File DAPPT.sol

```
    function startFunding (uint256 _fundingStartBlock, uint256 _fundingStopBlock)
        isOwner external {
        if (isFunding) throw;
        if (_fundingStartBlock >= _fundingStopBlock) throw;
        if (block.number >= _fundingStartBlock) throw;

        fundingStartBlock = _fundingStartBlock;
        fundingStopBlock = _fundingStopBlock;
        isFunding = true;
    }
```

✓ The code meets the specification.

## Formal Verification Request 26

**If method completes, integer overflow would not happen.**

06, Aug 2019
22.31 ms

Line 319 in File DAPPT.sol

```
    //@CTK NO_OVERFLOW
```

Line 328-331 in File DAPPT.sol

```
    function stopFunding() isOwner external {
        if (!isFunding) throw;
        isFunding = false;
    }
```

✓ The code meets the specification.

## Formal Verification Request 27

**Buffer overflow / array index out of bound would never happen.**

06, Aug 2019
0.54 ms

Line 320 in File DAPPT.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 328-331 in File DAPPT.sol

```
    function stopFunding() isOwner external {
        if (!isFunding) throw;
        isFunding = false;
    }
```

✓ The code meets the specification.

## Formal Verification Request 28

**Method will not encounter an assertion failure.**

06, Aug 2019
0.61 ms

Line 321 in File DAPPT.sol

```
    //@CTK NO_ASF
```

Line 328-331 in File DAPPT.sol

```
    function stopFunding() isOwner external {
        if (!isFunding) throw;
        isFunding = false;
    }
```

✓ The code meets the specification.

## Formal Verification Request 29

**stopFunding**

📅 06, Aug 2019
⏱ 3.92 ms

Line 322-327 in File DAPPT.sol

```
    /*@CTK stopFunding
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post isFunding
      @post !__post.isFunding
     */
```

Line 328-331 in File DAPPT.sol

```
    function stopFunding() isOwner external {
        if (!isFunding) throw;
        isFunding = false;
    }
```

✅ The code meets the specification.

## Formal Verification Request 30

**If method completes, integer overflow would not happen.**

📅 06, Aug 2019
⏱ 22.09 ms

Line 334 in File DAPPT.sol

```
    //@CTK NO_OVERFLOW
```

Line 343-346 in File DAPPT.sol

```
    function setMigrateContract(address _newContractAddr) isOwner external {
        if (_newContractAddr == newContractAddr) throw;
        newContractAddr = _newContractAddr;
    }
```

✅ The code meets the specification.

## Formal Verification Request 31

**Buffer overflow / array index out of bound would never happen.**

📅 06, Aug 2019
⏱ 0.59 ms

Line 335 in File DAPPT.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 343-346 in File DAPPT.sol

```
343     function setMigrateContract(address _newContractAddr) isOwner external {
344         if (_newContractAddr == newContractAddr) throw;
345         newContractAddr = _newContractAddr;
346     }
```

✅ The code meets the specification.

## Formal Verification Request 32

### Method will not encounter an assertion failure.

📅 06, Aug 2019
⏱️ 0.6 ms

Line 336 in File DAPPT.sol

```
336     //@CTK NO_ASF
```

Line 343-346 in File DAPPT.sol

```
343     function setMigrateContract(address _newContractAddr) isOwner external {
344         if (_newContractAddr == newContractAddr) throw;
345         newContractAddr = _newContractAddr;
346     }
```

✅ The code meets the specification.

## Formal Verification Request 33

### setMigrateContract

📅 06, Aug 2019
⏱️ 3.46 ms

Line 337-342 in File DAPPT.sol

```
337     /*@CTK setMigrateContract
338       @tag assume_completion
339       @post msg.sender == ethFundDeposit
340       @post _newContractAddr != newContractAddr
341       @post __post.newContractAddr == _newContractAddr
342      */
```

Line 343-346 in File DAPPT.sol

```
343     function setMigrateContract(address _newContractAddr) isOwner external {
344         if (_newContractAddr == newContractAddr) throw;
345         newContractAddr = _newContractAddr;
346     }
```

✅ The code meets the specification.

## Formal Verification Request 34

**If method completes, integer overflow would not happen.**

📅 06, Aug 2019
⏱ 20.98 ms

Line 349 in File DAPPT.sol

```
    //@CTK NO_OVERFLOW
```

Line 358-361 in File DAPPT.sol

```
    function changeOwner(address _newFundDeposit) isOwner() external {
        if (_newFundDeposit == address(0x0)) throw;
        ethFundDeposit = _newFundDeposit;
    }
```

✓ The code meets the specification.

## Formal Verification Request 35

**Buffer overflow / array index out of bound would never happen.**

📅 06, Aug 2019
⏱ 0.53 ms

Line 350 in File DAPPT.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 358-361 in File DAPPT.sol

```
    function changeOwner(address _newFundDeposit) isOwner() external {
        if (_newFundDeposit == address(0x0)) throw;
        ethFundDeposit = _newFundDeposit;
    }
```

✓ The code meets the specification.

## Formal Verification Request 36

**Method will not encounter an assertion failure.**

📅 06, Aug 2019
⏱ 0.5 ms

Line 351 in File DAPPT.sol

```
    //@CTK NO_ASF
```

Line 358-361 in File DAPPT.sol

```
    function changeOwner(address _newFundDeposit) isOwner() external {
        if (_newFundDeposit == address(0x0)) throw;
        ethFundDeposit = _newFundDeposit;
    }
```

✓ The code meets the specification.

## Formal Verification Request 37

### changeOwner

06, Aug 2019
3.67 ms

Line 352-357 in File DAPPT.sol

```
    /*@CTK changeOwner
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post _newFundDeposit != address(0)
      @post __post.ethFundDeposit == _newFundDeposit
     */
```

Line 358-361 in File DAPPT.sol

```
    function changeOwner(address _newFundDeposit) isOwner() external {
        if (_newFundDeposit == address(0x0)) throw;
        ethFundDeposit = _newFundDeposit;
    }
```

✓ The code meets the specification.

## Source Code with CertiK Labels

File DAPPT.sol

```
/**
 *Submitted for verification at Etherscan.io on 2019-03-08
*/

pragma solidity ^0.4.12;

contract IMigrationContract {
    function migrate(address addr, uint256 dappt) returns (bool success);
}

/* taking ideas from FirstBlood token */
contract SafeMath {

    /* function assert(bool assertion) internal { */
    /*   if (!assertion) { */
    /*     throw; */
    /*   } */
    /* }      // assert no longer needed once solidity is on 0.4.10 */

    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath add"
      @post (x + y < x || x + y < y) == __reverted
      @post !__reverted -> __return == x + y
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
     */
    function safeAdd(uint256 x, uint256 y) internal returns(uint256) {
      uint256 z = x + y;
      assert((z >= x) && (z >= y));
      return z;
    }

    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath sub"
      @post (x < y) == __reverted
      @post !__reverted -> __return == x - y
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
     */
    function safeSubtract(uint256 x, uint256 y) internal returns(uint256) {
      assert(x >= y);
      uint256 z = x - y;
      return z;
    }

    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath mul"
      @post ((x == 0) && (((x * y) / x) != y)) == (__reverted)
      @post !__reverted -> __return == x * y
      @post !__reverted == !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
      */
    function safeMult(uint256 x, uint256 y) internal returns(uint256) {
      uint256 z = x * y;
      assert((x == 0)||(z/x == y));
      return z;
    }

}

contract Token {
    uint256 public totalSupply;
    function balanceOf(address _owner) constant returns (uint256 balance);
    function transfer(address _to, uint256 _value) returns (bool success);
    function transferFrom(address _from, address _to, uint256 _value) returns (bool
        success);
    function approve(address _spender, uint256 _value) returns (bool success);
    function allowance(address _owner, address _spender) constant returns (uint256
        remaining);
    event Transfer(address indexed _from, address indexed _to, uint256 _value);
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);
}


/*  ERC 20 token */
contract StandardToken is Token {

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK transfer
      @tag assume_completion
      @post _to != address(0)
      @post (balances[msg.sender] >= _value && _value > 0) -> success == true
      @post (balances[msg.sender] >= _value && _value > 0 && msg.sender != _to) -> (
          __post.balances[_to] == balances[_to] + _value)
      @post (balances[msg.sender] >= _value && _value > 0 && msg.sender != _to) -> (
          __post.balances[msg.sender] == balances[msg.sender] - _value)
      @post (balances[msg.sender] < _value || _value == 0) -> success == false
      @post (balances[msg.sender] < _value || _value == 0 || msg.sender == _to) ->
          __post.balances[_to] == balances[_to]
      @post (balances[msg.sender] < _value || _value == 0 || msg.sender == _to) ->
          __post.balances[msg.sender] == balances[msg.sender]
     */
    function transfer(address _to, uint256 _value) returns (bool success) {
      if (balances[msg.sender] >= _value && _value > 0) {
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        Transfer(msg.sender, _to, _value);
        return true;
      } else {
        return false;
      }
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK transferFrom
```

```
      @tag assume_completion
      @pre _to != address(0)
      @post (balances[_from] >= _value && allowed[_from][msg.sender] >= _value &&
          _value > 0) -> success == true
      @post (balances[_from] >= _value && allowed[_from][msg.sender] >= _value &&
          _value > 0) -> (__post.allowed[_from][msg.sender] == allowed[_from][msg.
          sender] - _value)
      @post (balances[_from] >= _value && allowed[_from][msg.sender] >= _value &&
          _value > 0 && _from != _to) -> (__post.balances[_from] == balances[_from] -
          _value)
      @post (balances[_from] >= _value && allowed[_from][msg.sender] >= _value &&
          _value > 0 && _from != _to) -> (__post.balances[_to] == balances[_to] +
          _value)
      @post (balances[_from] < _value || allowed[_from][msg.sender] < _value || _value
           == 0) -> success == false
      @post (balances[_from] < _value || allowed[_from][msg.sender] < _value || _value
           == 0) -> (__post.allowed[_from][msg.sender] == allowed[_from][msg.sender])
      @post (balances[_from] < _value || allowed[_from][msg.sender] < _value || _value
           == 0 || _from == _to) -> __post.balances[_to] == balances[_to]
      @post (balances[_from] < _value || allowed[_from][msg.sender] < _value || _value
           == 0 || _from == _to) -> __post.balances[_from] == balances[_from]
     */
    function transferFrom(address _from, address _to, uint256 _value) returns (bool
        success) {
      if (balances[_from] >= _value && allowed[_from][msg.sender] >= _value && _value
          > 0) {
        balances[_to] += _value;
        balances[_from] -= _value;
        allowed[_from][msg.sender] -= _value;
        Transfer(_from, _to, _value);
        return true;
      } else {
        return false;
      }
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK balanceOf
      @tag assume_completion
      @post balance == (balances[_owner])
     */
    function balanceOf(address _owner) constant returns (uint256 balance) {
        return balances[_owner];
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK approve
      @tag assume_completion
      @post (__post.allowed[msg.sender][_spender]) == (_value)
      @post (success) == (true)
     */
    function approve(address _spender, uint256 _value) returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK allowance
      @tag assume_completion
      @post (__reverted) == (false)
      @post (remaining) == (__post.allowed[_owner][_spender])
     */
    function allowance(address _owner, address _spender) constant returns (uint256
        remaining) {
      return allowed[_owner][_spender];
    }

    mapping (address => uint256) balances;
    mapping (address => mapping (address => uint256)) allowed;
}

contract DappToken is StandardToken, SafeMath {

    // metadata
    string public constant name = "Dapp Token";
    string public constant symbol = "DAPPT";
    uint256 public constant decimals = 18;
    string public version = "1.0";

    // contracts
    address public ethFundDeposit;      // deposit address for ETH for Dapp Team.
    address public newContractAddr;     // the new contract for dapp token updates;

    // crowdsale parameters
    bool    public isFunding;              // switched to true in operational state
    uint256 public fundingStartBlock;
    uint256 public fundingStopBlock;

    uint256 public currentSupply;       // current supply tokens for sell
    uint256 public tokenRaised = 0;     // the number of total sold token
    uint256 public tokenMigrated = 0;  // the number of total transferted token
    uint256 public tokenExchangeRate = 25000;        // 25000 Dapp tokens per 1 ETH

    // events
    event AllocateToken(address indexed _to, uint256 _value); // allocate token for
        private sale;
    event IssueToken(address indexed _to, uint256 _value); // issue token for public
        sale;
    event IncreaseSupply(uint256 _value);
    event DecreaseSupply(uint256 _value);
    event Migrate(address indexed _to, uint256 _value);

    // format decimals.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK formatDecimals
      @tag assume_completion
      @post __return == _value * 1000000000000000000
     */
    function formatDecimals(uint256 _value) internal returns (uint256 ) {
        return _value * 10 ** decimals;
    }

    // constructor
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK DappToken
      @post currentSupply <= totalSupply
      @post __post.ethFundDeposit == _ethFundDeposit
      @post __post.isFunding == false
      @post __post.fundingStartBlock == 0
      @post __post.fundingStopBlock == 0
      @post __post.currentSupply == _currentSupply * 1000000000000000000
      @post __post.currentSupply == 5000000000 * 1000000000000000000
      @post __post.balances[msg.sender] == __post.currentSupply
     */
    function DappToken(address _ethFundDeposit, uint256 _currentSupply) {
        ethFundDeposit = _ethFundDeposit;

        isFunding = false;                          //controls pre through crowdsale state
        fundingStartBlock = 0;
        fundingStopBlock = 0;

        currentSupply = formatDecimals(_currentSupply);
        totalSupply = formatDecimals(5000000000);
      balances[msg.sender] = totalSupply;
        if(currentSupply > totalSupply) throw;
    }

    modifier isOwner() { require(msg.sender == ethFundDeposit); _; }

    /// @dev set the token's tokenExchangeRate,
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setTokenExchangeRate
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post _tokenExchangeRate != 0
      @post _tokenExchangeRate != tokenExchangeRate
      @post __post.tokenExchangeRate == _tokenExchangeRate
     */
    function setTokenExchangeRate(uint256 _tokenExchangeRate) isOwner external {
        if (_tokenExchangeRate == 0) throw;
        if (_tokenExchangeRate == tokenExchangeRate) throw;

        tokenExchangeRate = _tokenExchangeRate;
    }

    /// @dev increase the token's supply
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK increaseSupply
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post currentSupply + (_value * 1000000000000000000) <= totalSupply
      @post __post.currentSupply == currentSupply + (_value * 1000000000000000000)
     */
    function increaseSupply (uint256 _value) isOwner external {
        uint256 value = formatDecimals(_value);
        if (value + currentSupply > totalSupply) throw;
        currentSupply = safeAdd(currentSupply, value);
        IncreaseSupply(value);
    }

    /// @dev decrease the token's supply
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK decreaseSupply
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post (_value * 1000000000000000000) + tokenRaised <= currentSupply
      @post __post.currentSupply == currentSupply - (_value * 1000000000000000000)
     */
    function decreaseSupply (uint256 _value) isOwner external {
        uint256 value = formatDecimals(_value);
        if (value + tokenRaised > currentSupply) throw;

        currentSupply = safeSubtract(currentSupply, value);
        DecreaseSupply(value);
    }

    /// @dev turn on the funding state
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK startFunding
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post !isFunding
      @post _fundingStartBlock < _fundingStopBlock
      @post block.number < _fundingStartBlock
      @post __post.fundingStartBlock == _fundingStartBlock
      @post __post.fundingStopBlock == _fundingStopBlock
      @post __post.isFunding
     */
    function startFunding (uint256 _fundingStartBlock, uint256 _fundingStopBlock)
        isOwner external {
        if (isFunding) throw;
        if (_fundingStartBlock >= _fundingStopBlock) throw;
        if (block.number >= _fundingStartBlock) throw;

        fundingStartBlock = _fundingStartBlock;
        fundingStopBlock = _fundingStopBlock;
        isFunding = true;
    }

    /// @dev turn off the funding state
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK stopFunding
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post isFunding
      @post !__post.isFunding
     */
    function stopFunding() isOwner external {
        if (!isFunding) throw;
        isFunding = false;
    }

    /// @dev set a new contract for recieve the tokens (for update contract)
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setMigrateContract
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post _newContractAddr != newContractAddr
      @post __post.newContractAddr == _newContractAddr
     */
    function setMigrateContract(address _newContractAddr) isOwner external {
        if (_newContractAddr == newContractAddr) throw;
        newContractAddr = _newContractAddr;
    }

    /// @dev set a new owner.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK changeOwner
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post _newFundDeposit != address(0)
      @post __post.ethFundDeposit == _newFundDeposit
     */
    function changeOwner(address _newFundDeposit) isOwner() external {
        if (_newFundDeposit == address(0x0)) throw;
        ethFundDeposit = _newFundDeposit;
    }

    /// sends the tokens to new contract
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK migrate
      @tag assume_completion
      @post !isFunding
      @post newContractAddr != address(0)
      @post balances[msg.sender] > 0
      @post __post.balances[msg.sender] == 0
     */
    function migrate() external {
        if(isFunding) throw;
        if(newContractAddr == address(0x0)) throw;

        uint256 tokens = balances[msg.sender];
        if (tokens == 0) throw;

        balances[msg.sender] = 0;

        IMigrationContract newContract = IMigrationContract(newContractAddr);
        if (!newContract.migrate(msg.sender, tokens)) throw;

        Migrate(msg.sender, tokens);           // log it
    }

    /// @dev sends ETH to Dapp team
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK transferETH
      @tag assume_completion
      @post msg.sender == ethFundDeposit
     */
    function transferETH() isOwner external {
        if (this.balance == 0) throw;
        if (!ethFundDeposit.send(this.balance)) throw;
    }

    /// @dev allocates Dapp tokens to pre-sell address.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK allocateToken
      @tag assume_completion
      @post msg.sender == ethFundDeposit
      @post _eth > 0
      @post _addr != address(0)
      @post (_eth * 1000000000000000000 * tokenExchangeRate) + tokenRaised <=
          currentSupply
      @post __post.tokenRaised == tokenRaised + (_eth * 1000000000000000000 *
          tokenExchangeRate)
      @post __post.balances[_addr] == balances[_addr] + (_eth * 1000000000000000000 *
          tokenExchangeRate)
     */
    function allocateToken (address _addr, uint256 _eth) isOwner external {
        if (_eth == 0) throw;
        if (_addr == address(0x0)) throw;

        uint256 tokens = safeMult(formatDecimals(_eth), tokenExchangeRate);
        if (tokens + tokenRaised > currentSupply) throw;

        tokenRaised = safeAdd(tokenRaised, tokens);
        balances[_addr] += tokens;

        AllocateToken(_addr, tokens); // logs token issued
    }

    /// buys the tokens
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK fallback
      @tag assume_completion
      @post isFunding
      @post msg.value > 0
      @post block.number >= fundingStartBlock
      @post block.number <= fundingStopBlock
      @post (msg.value * tokenExchangeRate) + tokenRaised <= currentSupply
      @post __post.tokenRaised == tokenRaised + (msg.value * tokenExchangeRate)
      @post __post.balances[msg.sender] == balances[msg.sender] + (msg.value *
          tokenExchangeRate)
     */
    function () payable {
        if (!isFunding) throw;
        if (msg.value == 0) throw;

        if (block.number < fundingStartBlock) throw;
        if (block.number > fundingStopBlock) throw;

        uint256 tokens = safeMult(msg.value, tokenExchangeRate);
        if (tokens + tokenRaised > currentSupply) throw;

        tokenRaised = safeAdd(tokenRaised, tokens);
        balances[msg.sender] += tokens;

        IssueToken(msg.sender, tokens); // logs token issued
    }
}
```