

CERTIK AUDIT REPORT FOR STASIS



Request Date: 2019-06-12
Revision Date: 2019-06-18
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	9
Formal Verification Results	10
How to read	10
Source Code with CertiK Labels	76

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 Stasis(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/>

Executive Summary

This report has been prepared as product of the Smart Contract Audit request by Stasis. This audit was conducted to discover issues and vulnerabilities in the source code of Stasis'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.

Jun 18, 2019



Type of Issues

CertiK smart label engine applied 100% covered 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.	1	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.	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.	1	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

Missing address checks in transfer functions.

(Note: The violations in the formal verification section of the report are for internal evaluation and are not indication of security issue in the client contracts. However we recommend replacing `assert` with `require` in `SafeMath` as did in OpenZeppelin’s latest `SafeMath` library.)

Manual Review Notes

Source Code SHA-256 Checksum

- **EURSToken.sol**

953ed4a23199ca0e00bb69bf1a605e96c9fed50a3d279aa1126efef958d12131

Summary

CertiK was chosen by Stasis to audit the design and implementation of its EURS smart contract. To ensure comprehensive protection, the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Recommendations

SafeMath:

1. (INFO) Recommend avoiding use of magic number in `safeAdd()` & `safeMul()`. This can be achieved by performing the math operation first and then assert the result.
 - Example for `safeAdd()`: `z = x + y; assert(z >= x);`
 - Example for `safeMul()`: `z = x * y; assert(z / x == y);`
2. (INFO) Recommend removing `return` statements from function `safeAdd()`, `safeSub()`, `safeMul()`, as the return value `uint256 z` has been specified in the functions already.
3. (INFO) Recommend complementing `SafeDiv()` to be consistent with the `SafeMath` library for future contract update.

AbstractToken:

1. (IMPORTANT) Missing address check in function `transfer (address _to, uint256 _value)` and `function transferFrom (address _from, address _to, uint256 _value)`, which may lead to possible value loss. Same for `EURSToken`, where the address check is missing in `function approve (address _spender, uint256 _value)`.
2. (INFO) Recommend using `require` statement at line 197, 198, 218, 220, 225 with error message provided.

EURSToken

1. (IMPORTANT) When the contract is initialized the `EURSToken()` has no checking for the `_feeCollector` address.
2. (IMPORTANT) Missing address check in `function setOwner`, `function setFeeCollector`, `function setDelegate`, `function approve`, and `transfer` functions.
3. (INFO) Recommend checking whether the `_from` address recovered from `ecrecover()` is zero address.
4. (INFO) `function transfer`: line 443/445 use `require` statement instead of `if` statement. Same for `function transferFrom`, `createTokens`, `burnTokens`, `freezeTransfers`, `unfreezeTransfers`.
5. (INFO) Recommend emitting error logs to transfer condition failure.
6. (INFO) Recommend complementing error messages to current `require()` statements.
7. (INFO) Recommend adding a deadline timestamp field besides `nonce` to the signature for `delegatedTransfer`.
8. (INFO) Inconsistent use of `SafeMath` in the `delegatedTransfer()` for the increment of nonce `nonces [_from] = _nonce + 1`, which may lead to integer overflow.
9. (INFO) Recommend following EIP-712 for the implementation of data signing and delegate transfer. Reference: [EIP-712](#).
10. (INFO) The EURS contract employs various roles and features such as `Owner`, `Delegator`, `FeeCollector` to meet its business requirements. We recommend:
 - Providing PAUSING functionality to be able to assist in legal investigation and auditing, as well as emergencies.
 - Splitting the responsibility into single aspects to minimize negative effects under special circumstances. For example:
 - `SupplyController`: `createToken` & `burnToken` operations
 - `RegularCompliance`: `freeze` & `unfreeze` operations
 - `Owner`: Administrative operations such as role transfer
11. (INFO) Recommend extracting the following checking logic in transfer functions as separate internal function/modifier to improve readability and testability:
 - `isTransferBlocked(address _from, address _to): (addressFlags [_from] | addressFlags [_to]) & BLACK_LIST_FLAG == BLACK_LIST_FLAG)`
 - `getTransferFee(address _from, address _to, uint256 _value): (addressFlags [_from] | addressFlags [_to]) & ZERO_FEE_FLAG == ZERO_FEE_FLAG ? 0 : calculateFee(_value)`
12. (INFO) Recommend use `onlyOwner` modifier in place of `require (msg.sender == owner)` and provide error message for `require()`.
13. (IMPORTANT) Recommend using pull model instead of push model for the transfer of ownership & delegated role to reduce the risk of manual mistake.

```
address indexed owner;
address indexed proposedOwner;
function proposeNewOwner(address newOwner) isOwner public {
    require(newOwner != address(0), ...);
    proposedOwner = newOwner;
    // emit LogOwnerTransferProposed ...
}
function claimOwnership() public {
    require(msg.sender == proposedOwner, ...);
    owner = proposedOwner;
    proposedOwner = address(0);
    // emit LogOwnerTransferred ...
}
address indexed delegate;
address indexed proposedDelegate;
function proposeNewDelegate(address newDelegate) isOwner public {
    require(newDelegate != address(0), ...);
    require(newDelegate != delegate, ...);
    proposedDelegate = newDelegate;
    // emit LogDelegateRoleTransferProposed ...
}
function claimDelegate() public {
    require(msg.sender == proposedDelegate, ...);
    delegate = proposedDelegate;
    proposedDelegate = address(0);
    // emit LogDelegateRoleTransferred ...
}
```

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 10 in File EURSToken.sol

```
10 pragma solidity ^0.4.20;
```

! Version to compile has the following bug: 0.4.20: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClipped-ABIV2, 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: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClipped-ABIV2

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	<pre> 30 /*@CTK FAIL "transferFrom to same address" 31 @tag assume_completion 32 @pre from == to 33 @post __post.allowed[from][msg.sender] == 34 */ </pre>
Raw code location	Line 35-41 in File howtoread.sol
Raw code	<pre> 35 function transferFrom(address from, address to 36) { 37 balances[from] = balances[from].sub(tokens 38 allowed[from][msg.sender] = allowed[from][39 balances[to] = balances[to].add(tokens); 40 emit Transfer(from, to, tokens); 41 return true; </pre>
Counterexample	<div>  This code violates the specification </div> <div> <pre> 1 Counter Example: 2 Before Execution: 3 Input = { 4 from = 0x0 5 to = 0x0 6 tokens = 0x6c 7 } 8 This = 0 </pre> </div> <div> <pre> 52 } 53 balance: 0x0 54 } 55 } </pre> </div> <div> <pre> 57 After Execution: 58 Input = { 59 from = 0x0 60 to = 0x0 61 tokens = 0x6c </pre> </div>

Formal Verification Request 1

Method will not encounter an assertion failure.

 18, Jun 2019

 15.76 ms

Line 26 in File EURSToken.sol

26 `//@CTK FAIL NO_ASF`

Line 35-40 in File EURSToken.sol

```

35 function safeAdd (uint256 x, uint256 y)
36 pure internal
37 returns (uint256 z) {
38     assert (x <= MAX_UINT256 - y);
39     return x + y;
40 }
```

 This code violates the specification.


```


1 Counter Example:
2 Before Execution:
3   Input = {
4     x = 3
5     y = 254
6   }
7   This = 0
8   Internal = {
9     __has_assertion_failure = false
10    __has_buf_overflow = false
11    __has_overflow = false
12    __has_returned = false
13    __reverted = false
14    msg = {
15      "gas": 0,
16      "sender": 0,
17      "value": 0
18    }
19  }
20  Other = {
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25    z = 0
26  }
27  Address_Map = [
28    {
29      "key": "ALL_OTHERS",
30      "value": {
31        "contract_name": "SafeMath",
32        "balance": 0,
33        "contract": {
34          "MAX_UINT256": 0
35        }
36      }
37    }
38  ]
```

39
40 Function invocation is reverted.

Formal Verification Request 2

SafeMath add

 18, Jun 2019

 5.78 ms

Line 27-34 in File EURSToken.sol

```
27  /*@CTK "SafeMath add"
28     @tag spec
29     @pre (MAX_UINT256) == (0xFF)
30     @post (x > (MAX_UINT256 - y)) == __reverted
31     @post (!__reverted) -> z == (x + y)
32     @post (!__reverted) -> !__has_overflow
33     @post !(__has_buf_overflow)
34  */
```

Line 35-40 in File EURSToken.sol


```
35  function safeAdd (uint256 x, uint256 y)
36  pure internal
37  returns (uint256 z) {
38      assert (x <= MAX_UINT256 - y);
39      return x + y;
40  }
```

 The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

 18, Jun 2019

 15.21 ms

Line 49 in File EURSToken.sol

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

Line 57-62 in File EURSToken.sol

```
57  function safeSub (uint256 x, uint256 y)
58  pure internal
59  returns (uint256 z) {
60      assert (x >= y);
61      return x - y;
62  }
```

 This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          x = 0
```

```

5     y = 1
6   }
7   This = 0
8   Internal = {
9     __has_assertion_failure = false
10    __has_buf_overflow = false
11    __has_overflow = false
12    __has_returned = false
13    __reverted = false
14    msg = {
15      "gas": 0,
16      "sender": 0,
17      "value": 0
18    }
19  }
20  Other = {
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25    z = 0
26  }
27  Address_Map = [
28    {
29      "key": "ALL_OTHERS",
30      "value": {
31        "contract_name": "SafeMath",
32        "balance": 0,
33        "contract": {
34          "MAX_UINT256": 0
35        }
36      }
37    }
38  ]
39
40  Function invocation is reverted.

```

Formal Verification Request 4

SafeMath sub



18, Jun 2019



1.35 ms

Line 50-56 in File EURSToken.sol

```

50  /*@CTK "SafeMath sub"
51    @tag spec
52    @post (x < y) == (__reverted)
53    @post (!__reverted) -> (z == (x - y))
54    @post (!__reverted) -> (!__has_overflow)
55    @post !(__has_buf_overflow)
56  */

```

Line 57-62 in File EURSToken.sol

```

57  function safeSub (uint256 x, uint256 y)
58  pure internal

```

```

59  returns (uint256 z) {
60      assert (x >= y);
61      return x - y;
62  }


```

✓ The code meets the specification.

Formal Verification Request 5

Method will not encounter an assertion failure.

 18, Jun 2019

 21.91 ms

Line 71 in File EURSToken.sol

```

71  //@CTK FAIL NO_ASF

```

Line 81-87 in File EURSToken.sol

```

81  function safeMul (uint256 x, uint256 y)
82  pure internal
83  returns (uint256 z) {
84      if (y == 0) return 0; // Prevent division by zero at the next line
85      assert (x <= MAX_UINT256 / y);
86      return x * y;
87  }

```

✗ This code violates the specification.

```

1  Counter Example:
2  Before Execution:
3      Input = {
4          x = 255
5          y = 1
6      }
7      This = 0
8      Internal = {
9          __has_assertion_failure = false
10         __has_buf_overflow = false
11         __has_overflow = false
12         __has_returned = false
13         __reverted = false
14         msg = {
15             "gas": 0,
16             "sender": 0,
17             "value": 0
18         }
19     }
20     Other = {
21         block = {
22             "number": 0,
23             "timestamp": 0
24         }
25         z = 0
26     }
27     Address_Map = [
28         {

```



```


29     "key": "ALL_OTHERS",
30     "value": {
31         "contract_name": "SafeMath",
32         "balance": 0,
33         "contract": {
34             "MAX_UINT256": 254
35         }
36     }
37 }
38 ]
39
40 Function invocation is reverted.

```

Formal Verification Request 6

SafeMath mul

 18, Jun 2019

 17.92 ms

Line 72-80 in File EURSToken.sol

```

72  /*@CTK "SafeMath mul"
73      @tag spec
74      @pre (MAX_UINT256) == (0xFF)
75      @post (y == 0) -> (z == 0)
76      @post (x > (MAX_UINT256 / y)) == (__reverted)
77      @post (!__reverted) -> (z == (x * y))
78      @post (!__reverted) -> (!__has_overflow)
79      @post !(__has_buf_overflow)
80  */

```

Line 81-87 in File EURSToken.sol

```

81  function safeMul (uint256 x, uint256 y)
82  pure internal
83  returns (uint256 z) {
84      if (y == 0) return 0; // Prevent division by zero at the next line
85      assert (x <= MAX_UINT256 / y);
86      return x * y;
87  }


```

 The code meets the specification.

Formal Verification Request 7

balanceOf

 18, Jun 2019

 5.87 ms

Line 205-209 in File EURSToken.sol

```

205  /*@CTK "balanceOf"
206      @tag spec
207      @tag assume_completion

```

```

208  @post balance == accounts [_owner]
209  */

```

Line 210-212 in File EURSToken.sol

```

210  function balanceOf (address _owner) public view returns (uint256 balance) {
211      return accounts [_owner];
212  }

```

✓ The code meets the specification.

Formal Verification Request 8

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 98.15 ms

Line 221 in File EURSToken.sol

```

221  //@CTK FAIL NO_OVERFLOW

```

Line 238-248 in File EURSToken.sol

```

238  function transfer (address _to, uint256 _value)
239  public payable returns (bool success) {
240      uint256 fromBalance = accounts [msg.sender];
241      if (fromBalance < _value) return false;
242      if (_value > 0 && msg.sender != _to) {
243          accounts [msg.sender] = safeSub (fromBalance, _value);
244          accounts [_to] = safeAdd (accounts [_to], _value);
245      }
246      Transfer (msg.sender, _to, _value);
247      return true;
248  }

```

✗ This code violates the specification.

```

1  Counter Example:
2  Before Execution:
3      Input = {
4          _to = 1
5          _value = 128
6      }
7      This = 0
8      Internal = {
9          __has_assertion_failure = false
10         __has_buf_overflow = false
11         __has_overflow = false
12         __has_returned = false
13         __reverted = false
14         msg = {
15             "gas": 0,
16             "sender": 0,
17             "value": 0
18         }
19     }
20     Other = {
21         block = {

```

```

22     "number": 0,
23     "timestamp": 0
24 }
25 success = false
26 }
27 Address_Map = [
28 {
29     "key": 0,
30     "value": {
31         "contract_name": "AbstractToken",
32         "balance": 0,
33         "contract": {
34             "accounts": [
35                 {
36                     "key": 0,
37                     "value": 128
38                 },
39                 {
40                     "key": 1,
41                     "value": 0
42                 },
43                 {
44                     "key": "ALL_OTHERS",
45                     "value": 1
46                 }
47             ],
48             "allowances": [
49                 {
50                     "key": "ALL_OTHERS",
51                     "value": [
52                         {
53                             "key": "ALL_OTHERS",
54                             "value": 0
55                         }
56                     ]
57                 }
58             ],
59             "MAX_UINT256": 0
60         }
61     }
62 },
63 {
64     "key": "ALL_OTHERS",
65     "value": "EmptyAddress"
66 }
67 ]

```

69 After Execution:

```

70 Input = {
71     _to = 1
72     _value = 128
73 }
74 This = 0
75 Internal = {
76     __has_assertion_failure = false
77     __has_buf_overflow = false
78     __has_overflow = true
79     __has_returned = true

```


```


80     __reverted = false
81     msg = {
82         "gas": 0,
83         "sender": 0,
84         "value": 0
85     }
86 }
87 Other = {
88     block = {
89         "number": 0,
90         "timestamp": 0
91     }
92     fromBalance = 128
93     success = true
94 }
95 Address_Map = [
96     {
97         "key": 0,
98         "value": {
99             "contract_name": "AbstractToken",
100             "balance": 0,
101             "contract": {
102                 "accounts": [
103                     {
104                         "key": 0,
105                         "value": 0
106                     },
107                     {
108                         "key": 1,
109                         "value": 128
110                     },
111                     {
112                         "key": "ALL_OTHERS",
113                         "value": 1
114                     }
115                 ],
116                 "allowances": [
117                     {
118                         "key": "ALL_OTHERS",
119                         "value": [
120                             {
121                                 "key": "ALL_OTHERS",
122                                 "value": 0
123                             }
124                         ]
125                     }
126                 ],
127                 "MAX_UINT256": 0
128             }
129         }
130     },
131     {
132         "key": "ALL_OTHERS",
133         "value": "EmptyAddress"
134     }
135 ]

```

Formal Verification Request 9

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

 18, Jun 2019

 1.29 ms

Line 222 in File EURSToken.sol

222 `//@CTK NO_BUF_OVERFLOW`


Line 238-248 in File EURSToken.sol


```
238 function transfer (address _to, uint256 _value)
239 public payable returns (bool success) {
240     uint256 fromBalance = accounts [msg.sender];
241     if (fromBalance < _value) return false;
242     if (_value > 0 && msg.sender != _to) {
243         accounts [msg.sender] = safeSub (fromBalance, _value);
244         accounts [_to] = safeAdd (accounts [_to], _value);
245     }
246     Transfer (msg.sender, _to, _value);
247     return true;
248 }
```

 The code meets the specification.

Formal Verification Request 10

Method will not encounter an assertion failure.

 18, Jun 2019

 14.78 ms

Line 223 in File EURSToken.sol

223 `//@CTK FAIL NO_ASF`

Line 238-248 in File EURSToken.sol

```
238 function transfer (address _to, uint256 _value)
239 public payable returns (bool success) {
240     uint256 fromBalance = accounts [msg.sender];
241     if (fromBalance < _value) return false;
242     if (_value > 0 && msg.sender != _to) {
243         accounts [msg.sender] = safeSub (fromBalance, _value);
244         accounts [_to] = safeAdd (accounts [_to], _value);
245     }
246     Transfer (msg.sender, _to, _value);
247     return true;
248 }
```

 This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3   Input = {
4     _to = 4
5     _value = 97
```

```

6   }
7   This = 0
8   Internal = {
9       __has_assertion_failure = false
10      __has_buf_overflow = false
11      __has_overflow = false
12      __has_returned = false
13      __reverted = false
14      msg = {
15          "gas": 0,
16          "sender": 0,
17          "value": 0
18      }
19  }
20  Other = {
21      block = {
22          "number": 0,
23          "timestamp": 0
24      }
25      success = false
26  }
27  Address_Map = [
28      {
29          "key": 0,
30          "value": {
31              "contract_name": "AbstractToken",
32              "balance": 0,
33              "contract": {
34                  "accounts": [
35                      {
36                          "key": 0,
37                          "value": 225
38                      },
39                      {
40                          "key": 4,
41                          "value": 64
42                      },
43                      {
44                          "key": "ALL_OTHERS",
45                          "value": 0
46                      }
47                  ],
48                  "allowances": [
49                      {
50                          "key": "ALL_OTHERS",
51                          "value": [
52                              {
53                                  "key": "ALL_OTHERS",
54                                  "value": 4
55                              }
56                          ]
57                      }
58                  ],
59                  "MAX_UINT256": 128
60              }
61          }
62      },
63      {

```

```


64     "key": "ALL_OTHERS",
65     "value": "EmptyAddress"
66   }
67 ]
68
69 Function invocation is reverted.

```

Formal Verification Request 11

transfer error

 18, Jun 2019

 110.33 ms

Line 224-230 in File EURSToken.sol

```

224  /*@CTK "transfer error"
225    @tag spec
226    @tag assume_completion
227    @post (accounts[msg.sender] < _value) -> (success == false)
228    @post (accounts[msg.sender] >= _value && msg.sender != _to) -> (__post.accounts[
      msg.sender] == accounts[msg.sender] - _value) && (__post.accounts[_to] ==
      accounts[_to] + _value)
229    @post (accounts[msg.sender] < _value || _value == 0 || msg.sender == _to) -> (
      __post.accounts[msg.sender] == accounts[msg.sender]) && (__post.accounts[_to]
      == accounts[_to])
230  */

```

Line 238-248 in File EURSToken.sol

```

238  function transfer (address _to, uint256 _value)
239  public payable returns (bool success) {
240    uint256 fromBalance = accounts [msg.sender];
241    if (fromBalance < _value) return false;
242    if (_value > 0 && msg.sender != _to) {
243      accounts [msg.sender] = safeSub (fromBalance, _value);
244      accounts [_to] = safeAdd (accounts [_to], _value);
245    }
246    Transfer (msg.sender, _to, _value);
247    return true;
248  }


```

 The code meets the specification.

Formal Verification Request 12

transfer

 18, Jun 2019

 110.99 ms

Line 231-237 in File EURSToken.sol

```

231  /*@CTK "transfer"
232    @tag spec
233    @tag assume_completion

```

```

234 @post (accounts[msg.sender] < _value) -> (success == false)
235 @post (accounts[msg.sender] >= _value && msg.sender != _to) -> (__post.accounts[
    msg.sender] == accounts[msg.sender] - _value) && (__post.accounts[_to] ==
    accounts[_to] + _value)
236 @post (accounts[msg.sender] < _value || _value == 0 || msg.sender == _to) -> (
    __post.accounts[msg.sender] == accounts[msg.sender]) && (__post.accounts[_to]
    == accounts[_to])
237 */

```

Line 238-248 in File EURSToken.sol

```

238 function transfer (address _to, uint256 _value)
239 public payable returns (bool success) {
240     uint256 fromBalance = accounts [msg.sender];
241     if (fromBalance < _value) return false;
242     if (_value > 0 && msg.sender != _to) {
243         accounts [msg.sender] = safeSub (fromBalance, _value);
244         accounts [_to] = safeAdd (accounts [_to], _value);
245     }
246     Transfer (msg.sender, _to, _value);
247     return true;
248 }

```

✓ The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 128.59 ms

Line 259 in File EURSToken.sol

```

259 // @CTK FAIL NO_OVERFLOW

```

Line 272-288 in File EURSToken.sol

```

272 function transferFrom (address _from, address _to, uint256 _value)
273 public payable returns (bool success) {
274     uint256 spenderAllowance = allowances [_from][msg.sender];
275     if (spenderAllowance < _value) return false;
276     uint256 fromBalance = accounts [_from];
277     if (fromBalance < _value) return false;
278
279     allowances [_from][msg.sender] =
280         safeSub (spenderAllowance, _value);
281
282     if (_value > 0 && _from != _to) {
283         accounts [_from] = safeSub (fromBalance, _value);
284         accounts [_to] = safeAdd (accounts [_to], _value);
285     }
286     Transfer (_from, _to, _value);
287     return true;
288 }

```

✗ This code violates the specification.


```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _from = 128
5     _to = 0
6     _value = 1
7   }
8   This = 0
9   Internal = {
10    __has_assertion_failure = false
11    __has_buf_overflow = false
12    __has_overflow = false
13    __has_returned = false
14    __reverted = false
15    msg = {
16      "gas": 0,
17      "sender": 0,
18      "value": 0
19    }
20  }
21  Other = {
22    block = {
23      "number": 0,
24      "timestamp": 0
25    }
26    success = false
27  }
28  Address_Map = [
29    {
30      "key": 0,
31      "value": {
32        "contract_name": "AbstractToken",
33        "balance": 0,
34        "contract": {
35          "accounts": [
36            {
37              "key": 0,
38              "value": 0
39            },
40            {
41              "key": 64,
42              "value": 16
43            },
44            {
45              "key": 32,
46              "value": 2
47            },
48            {
49              "key": "ALL_OTHERS",
50              "value": 128
51            }
52          ],
53          "allowances": [
54            {
55              "key": 128,
56              "value": [
57                {
58                  "key": 0,

```

```

59         "value": 128
60     },
61     {
62         "key": "ALL_OTHERS",
63         "value": 0
64     }
65 ]
66 },
67 {
68     "key": "ALL_OTHERS",
69     "value": [
70         {
71             "key": "ALL_OTHERS",
72             "value": 128
73         }
74     ]
75 }
76 ],
77 "MAX_UINT256": 0
78 }
79 }
80 },
81 {
82     "key": "ALL_OTHERS",
83     "value": "EmptyAddress"
84 }
85 ]

```

```

87 After Execution:
88     Input = {
89         _from = 128
90         _to = 0
91         _value = 1
92     }
93     This = 0
94     Internal = {
95         __has_assertion_failure = false
96         __has_buf_overflow = false
97         __has_overflow = true
98         __has_returned = true
99         __reverted = false
100     }
101     msg = {
102         "gas": 0,
103         "sender": 0,
104         "value": 0
105     }
106     Other = {
107         block = {
108             "number": 0,
109             "timestamp": 0
110         }
111         fromBalance = 128
112         spenderAllowance = 128
113         success = true
114     }
115     Address_Map = [
116         {

```

```

117     "key": 0,
118     "value": {
119         "contract_name": "AbstractToken",
120         "balance": 0,
121         "contract": {
122             "accounts": [
123                 {
124                     "key": 128,
125                     "value": 127
126                 },
127                 {
128                     "key": 0,
129                     "value": 1
130                 },
131                 {
132                     "key": 64,
133                     "value": 16
134                 },
135                 {
136                     "key": 32,
137                     "value": 2
138                 },
139                 {
140                     "key": "ALL_OTHERS",
141                     "value": 128
142                 }
143             ],
144             "allowances": [
145                 {
146                     "key": 128,
147                     "value": [
148                         {
149                             "key": 0,
150                             "value": 127
151                         },
152                         {
153                             "key": "ALL_OTHERS",
154                             "value": 0
155                         }
156                     ]
157                 },
158                 {
159                     "key": "ALL_OTHERS",
160                     "value": [
161                         {
162                             "key": "ALL_OTHERS",
163                             "value": 128
164                         }
165                     ]
166                 }
167             ],
168             "MAX_UINT256": 0
169         }
170     },
171 },
172 {
173     "key": "ALL_OTHERS",
174     "value": "EmptyAddress"


```

```
175     }  
176   ]
```

Formal Verification Request 14

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

 18, Jun 2019

 13.86 ms

Line 260 in File EURSToken.sol

```
260  //@CTK_NO_BUF_OVERFLOW
```

Line 272-288 in File EURSToken.sol


```
272  function transferFrom (address _from, address _to, uint256 _value)  
273  public payable returns (bool success) {  
274      uint256 spenderAllowance = allowances [_from][msg.sender];  
275      if (spenderAllowance < _value) return false;  
276      uint256 fromBalance = accounts [_from];  
277      if (fromBalance < _value) return false;  
278  
279      allowances [_from][msg.sender] =  
280          safeSub (spenderAllowance, _value);  
281  
282      if (_value > 0 && _from != _to) {  
283          accounts [_from] = safeSub (fromBalance, _value);  
284          accounts [_to] = safeAdd (accounts [_to], _value);  
285      }  
286      Transfer (_from, _to, _value);  
287      return true;  
288  }
```

 The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

 18, Jun 2019

 43.37 ms

Line 261 in File EURSToken.sol

```
261  //@CTK_FAIL_NO_ASF
```

Line 272-288 in File EURSToken.sol

```
272  function transferFrom (address _from, address _to, uint256 _value)  
273  public payable returns (bool success) {  
274      uint256 spenderAllowance = allowances [_from][msg.sender];  
275      if (spenderAllowance < _value) return false;  
276      uint256 fromBalance = accounts [_from];  
277      if (fromBalance < _value) return false;  
278  
279      allowances [_from][msg.sender] =
```

```

280     safeSub (spenderAllowance, _value);
281
282     if (_value > 0 && _from != _to) {
283         accounts [_from] = safeSub (fromBalance, _value);
284         accounts [_to] = safeAdd (accounts [_to], _value);
285     }
286     Transfer (_from, _to, _value);
287     return true;
288 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3     Input = {
4         _from = 0
5         _to = 64
6         _value = 1
7     }
8     This = 0
9     Internal = {
10         __has_assertion_failure = false
11         __has_buf_overflow = false
12         __has_overflow = false
13         __has_returned = false
14         __reverted = false
15         msg = {
16             "gas": 0,
17             "sender": 0,
18             "value": 0
19         }
20     }
21     Other = {
22         block = {
23             "number": 0,
24             "timestamp": 0
25         }
26         success = false
27     }
28     Address_Map = [
29         {
30             "key": 0,
31             "value": {
32                 "contract_name": "AbstractToken",
33                 "balance": 0,
34                 "contract": {
35                     "accounts": [
36                         {
37                             "key": 64,
38                             "value": 1
39                         },
40                         {
41                             "key": 128,
42                             "value": 32
43                         },
44                         {
45                             "key": 4,
46                             "value": 0
47                         },

```

```


48         {
49             "key": 8,
50             "value": 0
51         },
52         {
53             "key": "ALL_OTHERS",
54             "value": 128
55         }
56     ],
57     "allowances": [
58         {
59             "key": 0,
60             "value": [
61                 {
62                     "key": 128,
63                     "value": 0
64                 },
65                 {
66                     "key": 0,
67                     "value": 128
68                 },
69                 {
70                     "key": 4,
71                     "value": 128
72                 },
73                 {
74                     "key": 8,
75                     "value": 1
76                 },
77                 {
78                     "key": "ALL_OTHERS",
79                     "value": 64
80                 }
81             ]
82         },
83         {
84             "key": "ALL_OTHERS",
85             "value": [
86                 {
87                     "key": "ALL_OTHERS",
88                     "value": 128
89                 }
90             ]
91         }
92     ],
93     "MAX_UINT256": 1
94 }
95 }
96 },
97 {
98     "key": "ALL_OTHERS",
99     "value": "EmptyAddress"
100 }
101 ]


```

103 Function invocation is reverted.

Formal Verification Request 16

transferFrom

 18, Jun 2019

 182.42 ms

Line 262-271 in File EURSToken.sol

```

262  /*@CTK "transferFrom"
263     @tag spec
264     @tag assume_completion
265     @post (allowances[_from][msg.sender] < _value) -> (success == false)
266     @post (accounts [_from] < _value) -> (success == false)
267     @post (allowances[_from][msg.sender] >= _value && accounts [_from] >= _value) -> (
        __post.allowances[_from][msg.sender] == allowances[_from][msg.sender] - _value
    )
268     @post (allowances[_from][msg.sender] < _value || accounts [_from] < _value) -> (
        __post.allowances[_from][msg.sender] == allowances[_from][msg.sender])
269     @post (allowances[_from][msg.sender] >= _value && accounts [_from] >= _value &&
        _from != _to) -> (__post.accounts[_from] == accounts [_from] - _value) && (
        __post.accounts[_to] == accounts [_to] + _value)
270     @post (allowances[_from][msg.sender] < _value || accounts [_from] < _value ||
        _from == _to) -> (__post.accounts[_from] == accounts [_from]) && (__post.
        accounts[_to] == accounts [_to])
271  */

```

Line 272-288 in File EURSToken.sol

```

272  function transferFrom (address _from, address _to, uint256 _value)
273  public payable returns (bool success) {
274      uint256 spenderAllowance = allowances [_from][msg.sender];
275      if (spenderAllowance < _value) return false;
276      uint256 fromBalance = accounts [_from];
277      if (fromBalance < _value) return false;
278
279      allowances [_from][msg.sender] =
280          safeSub (spenderAllowance, _value);
281
282      if (_value > 0 && _from != _to) {
283          accounts [_from] = safeSub (fromBalance, _value);
284          accounts [_to] = safeAdd (accounts [_to], _value);
285      }
286      Transfer (_from, _to, _value);
287      return true;
288  }


```

 The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

 18, Jun 2019

 7.75 ms

Line 298 in File EURSToken.sol

298 // @CTK_NO_OVERFLOW

Line 306-312 in File EURSToken.sol

```
306 function approve (address _spender, uint256 _value)
307 public payable returns (bool success) {
308     allowances [msg.sender][_spender] = _value;
309     Approval (msg.sender, _spender, _value);
310
311     return true;
312 }
```

✓ The code meets the specification.

Formal Verification Request 18

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



18, Jun 2019



0.3 ms

Line 299 in File EURSToken.sol

299 // @CTK_NO_BUF_OVERFLOW

Line 306-312 in File EURSToken.sol

```
306 function approve (address _spender, uint256 _value)
307 public payable returns (bool success) {
308     allowances [msg.sender][_spender] = _value;
309     Approval (msg.sender, _spender, _value);
310
311     return true;
312 }
```

✓ The code meets the specification.

Formal Verification Request 19

Method will not encounter an assertion failure.



18, Jun 2019



0.34 ms

Line 300 in File EURSToken.sol

300 // @CTK_NO_ASF

Line 306-312 in File EURSToken.sol


```
306 function approve (address _spender, uint256 _value)
307 public payable returns (bool success) {
308     allowances [msg.sender][_spender] = _value;
309     Approval (msg.sender, _spender, _value);
310
311     return true;
312 }
```

✓ The code meets the specification.

Formal Verification Request 20

approve

 18, Jun 2019

 1.23 ms

Line 301-305 in File EURSToken.sol

```
301  /*@CTK "approve"
302     @tag spec
303     @tag assume_completion
304     @post (__post.allowances[msg.sender][_spender]) == (_value)
305  */
```

Line 306-312 in File EURSToken.sol


```
306  function approve (address _spender, uint256 _value)
307  public payable returns (bool success) {
308      allowances [msg.sender][_spender] = _value;
309      Approval (msg.sender, _spender, _value);
310
311      return true;
312  }
```

 The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

 18, Jun 2019

 4.57 ms

Line 325 in File EURSToken.sol

```
325  //@CTK NO_OVERFLOW
```

Line 333-336 in File EURSToken.sol


```
333  function allowance (address _owner, address _spender)
334  public view returns (uint256 remaining) {
335      return allowances [_owner][_spender];
336  }
```

 The code meets the specification.

Formal Verification Request 22

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

 18, Jun 2019

 0.28 ms

Line 326 in File EURSToken.sol

```
326  //@CTK NO_BUF_OVERFLOW
```

Line 333-336 in File EURSToken.sol


```
333     function allowance (address _owner, address _spender)
334     public view returns (uint256 remaining) {
335         return allowances [_owner][_spender];
336     }
```

✓ The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

 18, Jun 2019

 0.28 ms

Line 327 in File EURSToken.sol

```
327     //@CTK NO_ASF
```

Line 333-336 in File EURSToken.sol


```
333     function allowance (address _owner, address _spender)
334     public view returns (uint256 remaining) {
335         return allowances [_owner][_spender];
336     }
```

✓ The code meets the specification.

Formal Verification Request 24

allowance

 18, Jun 2019

 0.32 ms

Line 328-332 in File EURSToken.sol

```
328     /*@CTK "allowance"
329         @tag spec
330         @tag assume_completion
331         @post (remaining) == (allowances[_owner][_spender])
332     */
```


Line 333-336 in File EURSToken.sol


```
333     function allowance (address _owner, address _spender)
334     public view returns (uint256 remaining) {
335         return allowances [_owner][_spender];
336     }
```

✓ The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

 18, Jun 2019

 32.8 ms

Line 449 in File EURSToken.sol

449 `//@CTK NO_OVERFLOW`


Line 461-469 in File EURSToken.sol


```
461 function EURSToken (address _feeCollector) public {
462     fixedFee = DEFAULT_FEE;
463     minVariableFee = 0;
464     maxVariableFee = 0;
465     variableFeeNumerator = 0;
466
467     owner = msg.sender;
468     feeCollector = _feeCollector;
469 }
```

 The code meets the specification.

Formal Verification Request 26

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

 18, Jun 2019

 0.41 ms

Line 450 in File EURSToken.sol

450 `//@CTK NO_BUF_OVERFLOW`

Line 461-469 in File EURSToken.sol


```
461 function EURSToken (address _feeCollector) public {
462     fixedFee = DEFAULT_FEE;
463     minVariableFee = 0;
464     maxVariableFee = 0;
465     variableFeeNumerator = 0;
466
467     owner = msg.sender;
468     feeCollector = _feeCollector;
469 }
```

 The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

 18, Jun 2019

 0.41 ms

Line 451 in File EURSToken.sol

451 //CTK NO_ASF

Line 461-469 in File EURSToken.sol

```
461 function EURSToken (address _feeCollector) public {
462     fixedFee = DEFAULT_FEE;
463     minVariableFee = 0;
464     maxVariableFee = 0;
465     variableFeeNumerator = 0;
466
467     owner = msg.sender;
468     feeCollector = _feeCollector;
469 }
```

✓ The code meets the specification.

Formal Verification Request 28

EURSToken



18, Jun 2019



1.27 ms

Line 452-460 in File EURSToken.sol

```
452 /*CTK "EURSToken"
453     @tag assume_completion
454     @post __post.fixedFee == DEFAULT_FEE
455     @post __post.minVariableFee == 0
456     @post __post.maxVariableFee == 0
457     @post __post.variableFeeNumerator == 0
458     @post __post.owner == msg.sender
459     @post __post.feeCollector == _feeCollector
460 */
```

Line 461-469 in File EURSToken.sol

```
461 function EURSToken (address _feeCollector) public {
462     fixedFee = DEFAULT_FEE;
463     minVariableFee = 0;
464     maxVariableFee = 0;
465     variableFeeNumerator = 0;
466
467     owner = msg.sender;
468     feeCollector = _feeCollector;
469 }
```

✓ The code meets the specification.

Formal Verification Request 29

EURSToken totalSupply



18, Jun 2019



4.43 ms

Line 510-513 in File EURSToken.sol

```
510  /*@CTK "EURSToken totalSupply"
511      @tag assume_completion
512      @post __return == tokensCount
513  */
```

Line 514-516 in File EURSToken.sol


```
514  function totalSupply () public /*>IGNORE delegatable IGNORE<*/ view returns (uint256
    ) {
515      return tokensCount;
516  }
```

✓ The code meets the specification.

Formal Verification Request 30

EURSToken balanceOf

 18, Jun 2019

 20.79 ms

Line 525-528 in File EURSToken.sol

```
525  /*@CTK "EURSToken balanceOf"
526      @tag assume_completion
527      @post balance == accounts[_owner]
528  */
```

Line 529-532 in File EURSToken.sol


```
529  function balanceOf (address _owner)
530      public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 balance) {
531      return AbstractToken.balanceOf (_owner);
532  }
```

✓ The code meets the specification.

Formal Verification Request 31

If method completes, integer overflow would not happen.

 18, Jun 2019

 36.13 ms

Line 876 in File EURSToken.sol

```
876  //@CTK NO_OVERFLOW
```


Line 884-887 in File EURSToken.sol


```
884  function approve (address _spender, uint256 _value)
885      public /*>IGNORE delegatable IGNORE<*/ payable returns (bool success) {
886      return AbstractToken.approve (_spender, _value);
887  }
```

✓ The code meets the specification.

Formal Verification Request 32

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

 18, Jun 2019

 0.41 ms

Line 877 in File EURSToken.sol

877 `//@CTK NO_BUF_OVERFLOW`


Line 884-887 in File EURSToken.sol


```
884 function approve (address _spender, uint256 _value)
885 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool success) {
886     return AbstractToken.approve (_spender, _value);
887 }
```

 The code meets the specification.

Formal Verification Request 33

Method will not encounter an assertion failure.

 18, Jun 2019

 0.42 ms

Line 878 in File EURSToken.sol

878 `//@CTK NO_ASF`

Line 884-887 in File EURSToken.sol


```
884 function approve (address _spender, uint256 _value)
885 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool success) {
886     return AbstractToken.approve (_spender, _value);
887 }
```

 The code meets the specification.

Formal Verification Request 34

EURSToken approve

 18, Jun 2019

 1.81 ms

Line 879-883 in File EURSToken.sol

```
879 /*@CTK "EURSToken approve"
880 @tag spec
881 @tag assume_completion
882 @post (__post.allowances[msg.sender][_spender]) == (_value)
883 */
```

Line 884-887 in File EURSToken.sol


```
884 function approve (address _spender, uint256 _value)
885 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool success) {
886     return AbstractToken.approve (_spender, _value);
887 }
```

✓ The code meets the specification.

Formal Verification Request 35

If method completes, integer overflow would not happen.

 18, Jun 2019

 23.58 ms

Line 900 in File EURSToken.sol

```
900 // @CTK_NO_OVERFLOW
```

Line 908-911 in File EURSToken.sol


```
908 function allowance (address _owner, address _spender)
909 public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 remaining) {
910     return AbstractToken.allowance (_owner, _spender);
911 }
```

✓ The code meets the specification.

Formal Verification Request 36

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

 18, Jun 2019

 0.43 ms

Line 901 in File EURSToken.sol

```
901 // @CTK_NO_BUF_OVERFLOW
```

Line 908-911 in File EURSToken.sol


```
908 function allowance (address _owner, address _spender)
909 public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 remaining) {
910     return AbstractToken.allowance (_owner, _spender);
911 }
```

✓ The code meets the specification.

Formal Verification Request 37

Method will not encounter an assertion failure.

 18, Jun 2019

 0.42 ms

Line 902 in File EURSToken.sol

902 // @CTK NO_ASF

Line 908-911 in File EURSToken.sol

```

908 function allowance (address _owner, address _spender)
909 public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 remaining) {
910     return AbstractToken.allowance (_owner, _spender);
911 }


```

✓ The code meets the specification.

Formal Verification Request 38

EURSToken allowance

 18, Jun 2019

 0.43 ms

Line 903-907 in File EURSToken.sol

```

903 /*@CTK "EURSToken allowance"
904 @tag spec
905 @tag assume_completion
906 @post (remaining) == (allowances[_owner][_spender])
907 */

```

Line 908-911 in File EURSToken.sol

```

908 function allowance (address _owner, address _spender)
909 public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 remaining) {
910     return AbstractToken.allowance (_owner, _spender);
911 }


```

✓ The code meets the specification.

Formal Verification Request 39

If method completes, integer overflow would not happen.

 18, Jun 2019

 222.47 ms

Line 1149 in File EURSToken.sol

1149 // @CTK FAIL NO_OVERFLOW

Line 1159-1173 in File EURSToken.sol

```

1159 function createTokens (uint256 _value)
1160 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1161     require (msg.sender == owner);
1162
1163     if (_value > 0) {
1164         if (_value <= safeSub (MAX_TOKENS_COUNT, tokensCount)) {
1165             accounts [msg.sender] = safeAdd (accounts [msg.sender], _value);
1166             tokensCount = safeAdd (tokensCount, _value);
1167
1168             Transfer (address (0), msg.sender, _value);

```



```

1169
1170     return true;
1171 } else return false;
1172 } else return true;
1173 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _value = 66
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    __return = false
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25  }
26  Address_Map = [
27    {
28      "key": 0,
29      "value": {
30        "contract_name": "EURSToken",
31        "balance": 0,
32        "contract": {
33          "FEE_DENOMINATOR": 0,
34          "MAX_FEE_NUMERATOR": 0,
35          "MIN_FEE_NUMERATION": 0,
36          "MAX_TOKENS_COUNT": 135,
37          "DEFAULT_FEE": 0,
38          "BLACK_LIST_FLAG": 0,
39          "ZERO_FEE_FLAG": 0,
40          "owner": 0,
41          "feeCollector": 0,
42          "tokensCount": 7,
43          "frozen": false,
44          "nonces": [
45            {
46              "key": 0,
47              "value": 4
48            },
49            {
50              "key": 32,
51              "value": 16

```

```

52     },
53     {
54         "key": 8,
55         "value": 128
56     },
57     {
58         "key": "ALL_OTHERS",
59         "value": 190
60     }
61 ],
62 "fixedFee": 0,
63 "minVariableFee": 0,
64 "maxVariableFee": 0,
65 "variableFeeNumerator": 0,
66 "addressFlags": [
67     {
68         "key": 0,
69         "value": 0
70     },
71     {
72         "key": 16,
73         "value": 8
74     },
75     {
76         "key": "ALL_OTHERS",
77         "value": 190
78     }
79 ],
80 "delegate": 0,
81 "accounts": [
82     {
83         "key": 16,
84         "value": 0
85     },
86     {
87         "key": 0,
88         "value": 111
89     },
90     {
91         "key": 32,
92         "value": 0
93     },
94     {
95         "key": 8,
96         "value": 0
97     },
98     {
99         "key": "ALL_OTHERS",
100        "value": 190
101    }
102 ],
103 "allowances": [
104     {
105         "key": "ALL_OTHERS",
106         "value": [
107             {
108                 "key": "ALL_OTHERS",
109                 "value": 190

```

```

110         }
111     ]
112 }
113 ],
114     "MAX_UINT256": 32
115 }
116 }
117 },
118 {
119     "key": "ALL_OTHERS",
120     "value": "EmptyAddress"
121 }
122 ]
123
124 After Execution:
125     Input = {
126         _value = 66
127     }
128     This = 0
129     Internal = {
130         __has_assertion_failure = false
131         __has_buf_overflow = false
132         __has_overflow = true
133         __has_returned = true
134         __reverted = false
135         msg = {
136             "gas": 0,
137             "sender": 0,
138             "value": 0
139         }
140     }
141     Other = {
142         __return = true
143         block = {
144             "number": 0,
145             "timestamp": 0
146         }
147     }
148     Address_Map = [
149     {
150         "key": 0,
151         "value": {
152             "contract_name": "EURSToken",
153             "balance": 0,
154             "contract": {
155                 "FEE_DENOMINATOR": 0,
156                 "MAX_FEE_NUMERATOR": 0,
157                 "MIN_FEE_NUMERATION": 0,
158                 "MAX_TOKENS_COUNT": 135,
159                 "DEFAULT_FEE": 0,
160                 "BLACK_LIST_FLAG": 0,
161                 "ZERO_FEE_FLAG": 0,
162                 "owner": 0,
163                 "feeCollector": 0,
164                 "tokensCount": 73,
165                 "frozen": false,
166                 "nonces": [
167

```

```

168         "key": 0,
169         "value": 4
170     },
171     {
172         "key": 32,
173         "value": 16
174     },
175     {
176         "key": 8,
177         "value": 128
178     },
179     {
180         "key": "ALL_OTHERS",
181         "value": 190
182     }
183 ],
184 "fixedFee": 0,
185 "minVariableFee": 0,
186 "maxVariableFee": 0,
187 "variableFeeNumerator": 0,
188 "addressFlags": [
189     {
190         "key": 0,
191         "value": 0
192     },
193     {
194         "key": 16,
195         "value": 8
196     },
197     {
198         "key": "ALL_OTHERS",
199         "value": 190
200     }
201 ],
202 "delegate": 0,
203 "accounts": [
204     {
205         "key": 32,
206         "value": 0
207     },
208     {
209         "key": 0,
210         "value": 177
211     },
212     {
213         "key": 16,
214         "value": 0
215     },
216     {
217         "key": 8,
218         "value": 0
219     },
220     {
221         "key": "ALL_OTHERS",
222         "value": 190
223     }
224 ],
225 "allowances": [

```

```


226     {
227         "key": "ALL_OTHERS",
228         "value": [
229             {
230                 "key": "ALL_OTHERS",
231                 "value": 190
232             }
233         ]
234     }
235 ],
236 "MAX_UINT256": 32
237 }
238 }
239 },
240 {
241     "key": "ALL_OTHERS",
242     "value": "EmptyAddress"
243 }
244 ]

```

Formal Verification Request 40

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

 18, Jun 2019

 3.52 ms

Line 1150 in File EURSToken.sol

```
1150 // @CTK NO_BUF_OVERFLOW
```

Line 1159-1173 in File EURSToken.sol

```

1159 function createTokens (uint256 _value)
1160 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1161     require (msg.sender == owner);
1162
1163     if (_value > 0) {
1164         if (_value <= safeSub (MAX_TOKENS_COUNT, tokensCount)) {
1165             accounts [msg.sender] = safeAdd (accounts [msg.sender], _value);
1166             tokensCount = safeAdd (tokensCount, _value);
1167
1168             Transfer (address (0), msg.sender, _value);
1169
1170             return true;
1171         } else return false;
1172     } else return true;
1173 }


```

 The code meets the specification.

Formal Verification Request 41

Method will not encounter an assertion failure.

 18, Jun 2019

 352.92 ms

Line 1151 in File EURSToken.sol

```
1151 // @CTK FAIL NO_ASF
```

Line 1159-1173 in File EURSToken.sol

```
1159 function createTokens (uint256 _value)
1160 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1161     require (msg.sender == owner);
1162
1163     if (_value > 0) {
1164         if (_value <= safeSub (MAX_TOKENS_COUNT, tokensCount)) {
1165             accounts [msg.sender] = safeAdd (accounts [msg.sender], _value);
1166             tokensCount = safeAdd (tokensCount, _value);
1167
1168             Transfer (address (0), msg.sender, _value);
1169
1170             return true;
1171         } else return false;
1172     } else return true;
1173 }
```

✗ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3   Input = {
4     _value = 6
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    __return = false
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25  }
26  Address_Map = [
27    {
28      "key": 0,
29      "value": {
30        "contract_name": "EURSToken",
31        "balance": 0,
32        "contract": {
33          "FEE_DENOMINATOR": 0,
34          "MAX_FEE_NUMERATOR": 0,
35          "MIN_FEE_NUMERATION": 0,
```

```

36     "MAX_TOKENS_COUNT": 144,
37     "DEFAULT_FEE": 0,
38     "BLACK_LIST_FLAG": 0,
39     "ZERO_FEE_FLAG": 0,
40     "owner": 0,
41     "feeCollector": 0,
42     "tokensCount": 196,
43     "frozen": false,
44     "nonces": [
45         {
46             "key": 16,
47             "value": 8
48         },
49         {
50             "key": 0,
51             "value": 16
52         },
53         {
54             "key": 1,
55             "value": 64
56         },
57         {
58             "key": 129,
59             "value": 0
60         },
61         {
62             "key": 2,
63             "value": 0
64         },
65         {
66             "key": 4,
67             "value": 64
68         },
69         {
70             "key": "ALL_OTHERS",
71             "value": 196
72         }
73     ],
74     "fixedFee": 0,
75     "minVariableFee": 0,
76     "maxVariableFee": 0,
77     "variableFeeNumerator": 0,
78     "addressFlags": [
79         {
80             "key": 0,
81             "value": 0
82         },
83         {
84             "key": 128,
85             "value": 0
86         },
87         {
88             "key": 64,
89             "value": 0
90         },
91         {
92             "key": "ALL_OTHERS",
93             "value": 32

```

```

94     }
95   ],
96   "delegate": 0,
97   "accounts": [
98     {
99       "key": 32,
100      "value": 0
101    },
102    {
103      "key": 0,
104      "value": 226
105    },
106    {
107      "key": 129,
108      "value": 4
109    },
110    {
111      "key": 2,
112      "value": 64
113    },
114    {
115      "key": 4,
116      "value": 0
117    },
118    {
119      "key": 16,
120      "value": 1
121    },
122    {
123      "key": 6,
124      "value": 128
125    },
126    {
127      "key": 8,
128      "value": 32
129    },
130    {
131      "key": 128,
132      "value": 32
133    },
134    {
135      "key": "ALL_OTHERS",
136      "value": 196
137    }
138  ],
139  "allowances": [
140    {
141      "key": 0,
142      "value": [
143        {
144          "key": 16,
145          "value": 64
146        },
147        {
148          "key": 0,
149          "value": 0
150        }
151      ]

```



```


152         "key": 128,
153         "value": 32
154     },
155     {
156         "key": "ALL_OTHERS",
157         "value": 196
158     }
159 ]
160 },
161 {
162     "key": 64,
163     "value": [
164         {
165             "key": 0,
166             "value": 0
167         },
168         {
169             "key": "ALL_OTHERS",
170             "value": 64
171         }
172     ]
173 },
174 {
175     "key": "ALL_OTHERS",
176     "value": [
177         {
178             "key": "ALL_OTHERS",
179             "value": 32
180         }
181     ]
182 }
183 ],
184 "MAX_UINT256": 64
185 }
186 }
187 },
188 {
189     "key": "ALL_OTHERS",
190     "value": "EmptyAddress"
191 }
192 ]
193
194 Function invocation is reverted.

```

Formal Verification Request 42

EURSToken createTokens

 18, Jun 2019

 222.89 ms

Line 1152-1158 in File EURSToken.sol

```

1152  /*@CTK "EURSToken createTokens"
1153     @tag assume_completion
1154     @pre msg.sender == owner
1155     @post ((_value > 0) && (_value > MAX_TOKENS_COUNT - tokensCount)) -> !__return

```

```

1156 @post ((_value == 0) || (_value <= MAX_TOKENS_COUNT - tokensCount)) -> __return
1157 @post ((_value == 0) && (_value <= MAX_TOKENS_COUNT - tokensCount)) -> (__post.
    accounts[msg.sender] == accounts[msg.sender] + _value) && (__post.tokensCount
    == tokensCount + _value)
1158 */

```

Line 1159-1173 in File EURSToken.sol

```

1159 function createTokens (uint256 _value)
1160 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1161     require (msg.sender == owner);
1162
1163     if (_value > 0) {
1164         if (_value <= safeSub (MAX_TOKENS_COUNT, tokensCount)) {
1165             accounts [msg.sender] = safeAdd (accounts [msg.sender], _value);
1166             tokensCount = safeAdd (tokensCount, _value);
1167
1168             Transfer (address (0), msg.sender, _value);
1169
1170             return true;
1171         } else return false;
1172     } else return true;
1173 }


```

✓ The code meets the specification.

Formal Verification Request 43

If method completes, integer overflow would not happen.

 18, Jun 2019

 104.99 ms

Line 1180 in File EURSToken.sol

```

1180 //@CTK NO_OVERFLOW

```

Line 1190-1204 in File EURSToken.sol

```


1190 function burnTokens (uint256 _value)
1191 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1192     require (msg.sender == owner);
1193
1194     if (_value > 0) {
1195         if (_value <= accounts [msg.sender]) {
1196             accounts [msg.sender] = safeSub (accounts [msg.sender], _value);
1197             tokensCount = safeSub (tokensCount, _value);
1198
1199             Transfer (msg.sender, address (0), _value);
1200
1201             return true;
1202         } else return false;
1203     } else return true;
1204 }


```

✓ The code meets the specification.

Formal Verification Request 44

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

 18, Jun 2019

 7.81 ms

Line 1181 in File EURSToken.sol

1181 `//@CTK_NO_BUF_OVERFLOW`

Line 1190-1204 in File EURSToken.sol


```
1190 function burnTokens (uint256 _value)
1191 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1192     require (msg.sender == owner);
1193
1194     if (_value > 0) {
1195         if (_value <= accounts [msg.sender]) {
1196             accounts [msg.sender] = safeSub (accounts [msg.sender], _value);
1197             tokensCount = safeSub (tokensCount, _value);
1198
1199             Transfer (msg.sender, address (0), _value);
1200
1201             return true;
1202         } else return false;
1203     } else return true;
1204 }
```

 The code meets the specification.

Formal Verification Request 45

Method will not encounter an assertion failure.

 18, Jun 2019

 49.84 ms

Line 1182 in File EURSToken.sol

1182 `//@CTK_FAIL_NO_ASF`

Line 1190-1204 in File EURSToken.sol

```
1190 function burnTokens (uint256 _value)
1191 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1192     require (msg.sender == owner);
1193
1194     if (_value > 0) {
1195         if (_value <= accounts [msg.sender]) {
1196             accounts [msg.sender] = safeSub (accounts [msg.sender], _value);
1197             tokensCount = safeSub (tokensCount, _value);
1198
1199             Transfer (msg.sender, address (0), _value);
1200
1201             return true;
1202         } else return false;
1203     } else return true;
1204 }
```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _value = 32
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    __return = false
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25  }
26  Address_Map = [
27    {
28      "key": 0,
29      "value": {
30        "contract_name": "EURSToken",
31        "balance": 0,
32        "contract": {
33          "FEE_DENOMINATOR": 0,
34          "MAX_FEE_NUMERATOR": 0,
35          "MIN_FEE_NUMERATION": 0,
36          "MAX_TOKENS_COUNT": 0,
37          "DEFAULT_FEE": 0,
38          "BLACK_LIST_FLAG": 0,
39          "ZERO_FEE_FLAG": 0,
40          "owner": 0,
41          "feeCollector": 0,
42          "tokensCount": 0,
43          "frozen": false,
44          "nonces": [
45            {
46              "key": 4,
47              "value": 16
48            },
49            {
50              "key": 0,
51              "value": 32
52            },
53            {
54              "key": 2,
55              "value": 8
56            },
57            {

```

```
58         "key": "ALL_OTHERS",
59         "value": 128
60     }
61 ],
62     "fixedFee": 0,
63     "minVariableFee": 0,
64     "maxVariableFee": 0,
65     "variableFeeNumerator": 0,
66     "addressFlags": [
67     {
68         "key": 4,
69         "value": 0
70     },
71     {
72         "key": 2,
73         "value": 64
74     },
75     {
76         "key": 64,
77         "value": 0
78     },
79     {
80         "key": "ALL_OTHERS",
81         "value": 128
82     }
83 ],
84     "delegate": 0,
85     "accounts": [
86     {
87         "key": 2,
88         "value": 0
89     },
90     {
91         "key": 64,
92         "value": 16
93     },
94     {
95         "key": "ALL_OTHERS",
96         "value": 128
97     }
98 ],
99     "allowances": [
100     {
101         "key": "ALL_OTHERS",
102         "value": [
103         {
104             "key": "ALL_OTHERS",
105             "value": 128
106         }
107     ]
108     }
109 ],
110     "MAX_UINT256": 0
111 }
112 }
113 },
114 {
115     "key": "ALL_OTHERS",
```


```


116         "value": "EmptyAddress"
117     }
118 ]
119
120 Function invocation is reverted.

```

Formal Verification Request 46

EURSToken burnTokens

 18, Jun 2019

 285.3 ms

Line 1183-1189 in File EURSToken.sol

```

1183  /*@CTK "EURSToken burnTokens"
1184      @tag assume_completion
1185      @pre msg.sender == owner
1186      @post ((_value > 0) && (_value > accounts[msg.sender])) -> !__return
1187      @post ((_value == 0) || (_value <= accounts[msg.sender])) -> __return
1188      @post ((_value == 0) || (_value <= accounts[msg.sender])) -> (__post.accounts[msg
        .sender] == accounts[msg.sender] - _value) && (__post.tokensCount ==
        tokensCount - _value)
1189  */

```

Line 1190-1204 in File EURSToken.sol

```

1190  function burnTokens (uint256 _value)
1191  public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1192      require (msg.sender == owner);
1193
1194      if (_value > 0) {
1195          if (_value <= accounts[msg.sender]) {
1196              accounts[msg.sender] = safeSub (accounts[msg.sender], _value);
1197              tokensCount = safeSub (tokensCount, _value);
1198
1199              Transfer (msg.sender, address (0), _value);
1200
1201              return true;
1202          } else return false;
1203      } else return true;
1204  }


```

 The code meets the specification.

Formal Verification Request 47

If method completes, integer overflow would not happen.

 18, Jun 2019

 22.63 ms

Line 1209 in File EURSToken.sol

```

1209  //@CTK NO_OVERFLOW

```

Line 1217-1225 in File EURSToken.sol

```
1217 function freezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1218     require (msg.sender == owner);
1219
1220     if (!frozen) {
1221         frozen = true;
1222
1223         Freeze ();
1224     }
1225 }
```

✓ The code meets the specification.

Formal Verification Request 48

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



18, Jun 2019



0.51 ms

Line 1210 in File EURSToken.sol

```
1210 // @CTK_NO_BUF_OVERFLOW
```

Line 1217-1225 in File EURSToken.sol

```
1217 function freezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1218     require (msg.sender == owner);
1219
1220     if (!frozen) {
1221         frozen = true;
1222
1223         Freeze ();
1224     }
1225 }
```

✓ The code meets the specification.

Formal Verification Request 49

Method will not encounter an assertion failure.



18, Jun 2019



0.5 ms

Line 1211 in File EURSToken.sol

```
1211 // @CTK_NO_ASF
```

Line 1217-1225 in File EURSToken.sol

```
1217 function freezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1218     require (msg.sender == owner);
1219
1220     if (!frozen) {
1221         frozen = true;
1222
1223         Freeze ();
```

```
1224 }  
1225 }
```

✓ The code meets the specification.

Formal Verification Request 50

freezeTransfers

📅 18, Jun 2019

🕒 10.73 ms

Line 1212-1216 in File EURSToken.sol

```
1212 /*@CTK "freezeTransfers"  
1213 @tag assume_completion  
1214 @pre msg.sender == owner  
1215 @post __post.frozen  
1216 */
```

Line 1217-1225 in File EURSToken.sol

```
1217 function freezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {  
1218     require (msg.sender == owner);  
1219  
1220     if (!frozen) {  
1221         frozen = true;  
1222  
1223         Freeze ();  
1224     }  
1225 }
```

✓ The code meets the specification.

Formal Verification Request 51

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 21.61 ms

Line 1230 in File EURSToken.sol

```
1230 //@CTK NO_OVERFLOW
```

Line 1238-1246 in File EURSToken.sol


```
1238 function unfreezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {  
1239     require (msg.sender == owner);  
1240  
1241     if (frozen) {  
1242         frozen = false;  
1243  
1244         Unfreeze ();  
1245     }  
1246 }
```

✓ The code meets the specification.

Formal Verification Request 52

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

 18, Jun 2019

 0.5 ms

Line 1231 in File EURSToken.sol

1231 `//@CTK_NO_BUF_OVERFLOW`


Line 1238-1246 in File EURSToken.sol


```
1238 function unfreezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1239     require (msg.sender == owner);
1240
1241     if (frozen) {
1242         frozen = false;
1243
1244         Unfreeze ();
1245     }
1246 }
```

 The code meets the specification.

Formal Verification Request 53

Method will not encounter an assertion failure.

 18, Jun 2019

 0.5 ms

Line 1232 in File EURSToken.sol

1232 `//@CTK_NO_ASF`

Line 1238-1246 in File EURSToken.sol


```
1238 function unfreezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1239     require (msg.sender == owner);
1240
1241     if (frozen) {
1242         frozen = false;
1243
1244         Unfreeze ();
1245     }
1246 }
```

 The code meets the specification.

Formal Verification Request 54

unfreezeTransfers

 18, Jun 2019

 10.37 ms

Line 1233-1237 in File EURSToken.sol

```
1233  /*@CTK "unfreezeTransfers"
1234      @tag assume_completion
1235      @pre msg.sender == owner
1236      @post __post.frozen == false
1237  */
```

Line 1238-1246 in File EURSToken.sol


```
1238  function unfreezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1239      require (msg.sender == owner);
1240
1241      if (frozen) {
1242          frozen = false;
1243
1244          Unfreeze ();
1245      }
1246  }
```

✓ The code meets the specification.

Formal Verification Request 55

If method completes, integer overflow would not happen.

 18, Jun 2019

 17.97 ms

Line 1253 in File EURSToken.sol

```
1253  //@CTK NO_OVERFLOW
```

Line 1261-1265 in File EURSToken.sol


```
1261  function setOwner (address _newOwner) public {
1262      require (msg.sender == owner);
1263
1264      owner = _newOwner;
1265  }
```

✓ The code meets the specification.

Formal Verification Request 56

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

 18, Jun 2019

 0.47 ms

Line 1254 in File EURSToken.sol

```
1254  //@CTK NO_BUF_OVERFLOW
```


Line 1261-1265 in File EURSToken.sol


```
1261  function setOwner (address _newOwner) public {
1262      require (msg.sender == owner);
1263
1264      owner = _newOwner;
1265  }
```

✓ The code meets the specification.

Formal Verification Request 57

Method will not encounter an assertion failure.

 18, Jun 2019

 0.43 ms

Line 1255 in File EURSToken.sol

```
1255 // @CTK NO_ASF
```


Line 1261-1265 in File EURSToken.sol


```
1261 function setOwner (address _newOwner) public {
1262     require (msg.sender == owner);
1263
1264     owner = _newOwner;
1265 }
```

✓ The code meets the specification.

Formal Verification Request 58

unfreezeTransfers

 18, Jun 2019

 2.62 ms

Line 1256-1260 in File EURSToken.sol

```
1256 /* @CTK "unfreezeTransfers"
1257     @tag assume_completion
1258     @post msg.sender == owner
1259     @post __post.owner == _newOwner
1260 */
```

Line 1261-1265 in File EURSToken.sol


```
1261 function setOwner (address _newOwner) public {
1262     require (msg.sender == owner);
1263
1264     owner = _newOwner;
1265 }
```

✓ The code meets the specification.

Formal Verification Request 59

If method completes, integer overflow would not happen.

 18, Jun 2019

 16.58 ms

Line 1272 in File EURSToken.sol

1272 //CTK NO_OVERFLOW


Line 1280-1285 in File EURSToken.sol


```
1280 function setFeeCollector (address _newFeeCollector)
1281 public /*>IGNORE delegatable IGNORE<*/ payable {
1282     require (msg.sender == owner);
1283
1284     feeCollector = _newFeeCollector;
1285 }
```

✓ The code meets the specification.

Formal Verification Request 60

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

 18, Jun 2019

 0.42 ms

Line 1273 in File EURSToken.sol

1273 //CTK NO_BUF_OVERFLOW

Line 1280-1285 in File EURSToken.sol


```
1280 function setFeeCollector (address _newFeeCollector)
1281 public /*>IGNORE delegatable IGNORE<*/ payable {
1282     require (msg.sender == owner);
1283
1284     feeCollector = _newFeeCollector;
1285 }
```

✓ The code meets the specification.

Formal Verification Request 61

Method will not encounter an assertion failure.

 18, Jun 2019

 0.42 ms

Line 1274 in File EURSToken.sol

1274 //CTK NO_ASF


Line 1280-1285 in File EURSToken.sol


```
1280 function setFeeCollector (address _newFeeCollector)
1281 public /*>IGNORE delegatable IGNORE<*/ payable {
1282     require (msg.sender == owner);
1283
1284     feeCollector = _newFeeCollector;
1285 }
```

✓ The code meets the specification.

Formal Verification Request 62

setFeeCollector

 18, Jun 2019

 2.69 ms

Line 1275-1279 in File EURSToken.sol

```
1275  /*@CTK "setFeeCollector"
1276     @tag assume_completion
1277     @pre msg.sender == owner
1278     @post (__post.feeCollector) == (_newFeeCollector)
1279  */
```


Line 1280-1285 in File EURSToken.sol


```
1280  function setFeeCollector (address _newFeeCollector)
1281  public /*>IGNORE delegatable IGNORE<*/ payable {
1282      require (msg.sender == owner);
1283
1284      feeCollector = _newFeeCollector;
1285  }
```

 The code meets the specification.

Formal Verification Request 63

If method completes, integer overflow would not happen.

 18, Jun 2019

 5.55 ms

Line 1294 in File EURSToken.sol

```
1294  //@CTK NO_OVERFLOW
```

Line 1301-1303 in File EURSToken.sol


```
1301  function nonce (address _owner) public view /*>IGNORE delegatable IGNORE<*/ returns
1302      (uint256) {
1303      return nonces [_owner];
1304  }
```

 The code meets the specification.

Formal Verification Request 64

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

 18, Jun 2019

 0.35 ms

Line 1295 in File EURSToken.sol

```
1295  //@CTK NO_BUF_OVERFLOW
```

Line 1301-1303 in File EURSToken.sol


```
1301 function nonce (address _owner) public view /*>IGNORE delegatable IGNORE<*/ returns
      (uint256) {
1302     return nonces [_owner];
1303 }
```

✓ The code meets the specification.

Formal Verification Request 65

Method will not encounter an assertion failure.

 18, Jun 2019

 0.32 ms

Line 1296 in File EURSToken.sol

```
1296 //CTK_NO_ASF
```

Line 1301-1303 in File EURSToken.sol


```
1301 function nonce (address _owner) public view /*>IGNORE delegatable IGNORE<*/ returns
      (uint256) {
1302     return nonces [_owner];
1303 }
```

✓ The code meets the specification.

Formal Verification Request 66

nonce

 18, Jun 2019

 0.33 ms

Line 1297-1300 in File EURSToken.sol

```
1297 /*CTK nonce
1298     @tag assume_completion
1299     @post __return == nonces [_owner]
1300 */
```

Line 1301-1303 in File EURSToken.sol


```
1301 function nonce (address _owner) public view /*>IGNORE delegatable IGNORE<*/ returns
      (uint256) {
1302     return nonces [_owner];
1303 }
```

✓ The code meets the specification.

Formal Verification Request 67

If method completes, integer overflow would not happen.

 18, Jun 2019

 59.45 ms

Line 1313 in File EURSToken.sol

1313 `//@CTK_NO_OVERFLOW`

Line 1326-1343 in File EURSToken.sol


```
1326 function setFeeParameters (
1327     uint256 _fixedFee,
1328     uint256 _minVariableFee,
1329     uint256 _maxVariableFee,
1330     uint256 _variableFeeNumerator) public /*>IGNORE delegatable IGNORE<*/ payable {
1331     require (msg.sender == owner);
1332
1333     require (_minVariableFee <= _maxVariableFee);
1334     require (_variableFeeNumerator <= MAX_FEE_NUMERATOR);
1335
1336     fixedFee = _fixedFee;
1337     minVariableFee = _minVariableFee;
1338     maxVariableFee = _maxVariableFee;
1339     variableFeeNumerator = _variableFeeNumerator;
1340
1341     FeeChange (
1342         _fixedFee, _minVariableFee, _maxVariableFee, _variableFeeNumerator);
1343 }
```

✓ The code meets the specification.

Formal Verification Request 68

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

 18, Jun 2019

 35.88 ms

Line 1314 in File EURSToken.sol

1314 `//@CTK_NO_BUF_OVERFLOW`

Line 1326-1343 in File EURSToken.sol

```
1326 function setFeeParameters (
1327     uint256 _fixedFee,
1328     uint256 _minVariableFee,
1329     uint256 _maxVariableFee,
1330     uint256 _variableFeeNumerator) public /*>IGNORE delegatable IGNORE<*/ payable {
1331     require (msg.sender == owner);
1332
1333     require (_minVariableFee <= _maxVariableFee);
1334     require (_variableFeeNumerator <= MAX_FEE_NUMERATOR);
1335
1336     fixedFee = _fixedFee;
1337     minVariableFee = _minVariableFee;
1338     maxVariableFee = _maxVariableFee;
1339     variableFeeNumerator = _variableFeeNumerator;
1340
1341     FeeChange (
1342         _fixedFee, _minVariableFee, _maxVariableFee, _variableFeeNumerator);
1343 }
```

✓ The code meets the specification.

Formal Verification Request 69

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 35.97 ms

Line 1315 in File EURSToken.sol

1315 `//@CTK NO_ASF`

Line 1326-1343 in File EURSToken.sol

```
1326 function setFeeParameters (
1327     uint256 _fixedFee,
1328     uint256 _minVariableFee,
1329     uint256 _maxVariableFee,
1330     uint256 _variableFeeNumerator) public /*>IGNORE delegatable IGNORE<*/ payable {
1331     require (msg.sender == owner);
1332
1333     require (_minVariableFee <= _maxVariableFee);
1334     require (_variableFeeNumerator <= MAX_FEE_NUMERATOR);
1335
1336     fixedFee = _fixedFee;
1337     minVariableFee = _minVariableFee;
1338     maxVariableFee = _maxVariableFee;
1339     variableFeeNumerator = _variableFeeNumerator;
1340
1341     FeeChange (
1342         _fixedFee, _minVariableFee, _maxVariableFee, _variableFeeNumerator);
1343 }
```

✓ The code meets the specification.

Formal Verification Request 70

setFeeCollector

📅 18, Jun 2019

🕒 5.88 ms

Line 1316-1325 in File EURSToken.sol

```
1316 /*@CTK setFeeCollector
1317     @tag assume_completion
1318     @pre msg.sender == owner
1319     @pre _minVariableFee <= _maxVariableFee
1320     @pre _variableFeeNumerator <= MAX_FEE_NUMERATOR
1321     @post (__post.fixedFee) == (_fixedFee)
1322     @post (__post.minVariableFee) == (_minVariableFee)
1323     @post (__post.maxVariableFee) == (_maxVariableFee)
1324     @post (__post.variableFeeNumerator) == (_variableFeeNumerator)
1325 */
```


Line 1326-1343 in File EURSToken.sol

```

1326 function setFeeParameters (
1327     uint256 _fixedFee,
1328     uint256 _minVariableFee,
1329     uint256 _maxVariableFee,
1330     uint256 _variableFeeNumerator) public /*>IGNORE delegatable IGNORE<*/ payable {
1331     require (msg.sender == owner);
1332
1333     require (_minVariableFee <= _maxVariableFee);
1334     require (_variableFeeNumerator <= MAX_FEE_NUMERATOR);
1335
1336     fixedFee = _fixedFee;
1337     minVariableFee = _minVariableFee;
1338     maxVariableFee = _maxVariableFee;
1339     variableFeeNumerator = _variableFeeNumerator;
1340
1341     FeeChange (
1342         _fixedFee, _minVariableFee, _maxVariableFee, _variableFeeNumerator);
1343 }

```

✓ The code meets the specification.

Formal Verification Request 71

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 6.85 ms

Line 1350 in File EURSToken.sol

```

1350 // @CTK NO_OVERFLOW

```

Line 1360-1369 in File EURSToken.sol

```

1360 function getFeeParameters () public /*>IGNORE delegatable IGNORE<*/ view returns (
1361     uint256 _fixedFee,
1362     uint256 _minVariableFee,
1363     uint256 _maxVariableFee,
1364     uint256 _variableFeeNumerator) {
1365     _fixedFee = fixedFee;
1366     _minVariableFee = minVariableFee;
1367     _maxVariableFee = maxVariableFee;
1368     _variableFeeNumerator = variableFeeNumerator;
1369 }

```

✓ The code meets the specification.

Formal Verification Request 72

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

📅 18, Jun 2019

🕒 0.37 ms

Line 1351 in File EURSToken.sol

1351 //CTK NO_BUF_OVERFLOW

Line 1360-1369 in File EURSToken.sol

```
1360 function getFeeParameters () public /*>IGNORE delegatable IGNORE<*/ view returns (
1361     uint256 _fixedFee,
1362     uint256 _minVariableFee,
1363     uint256 _maxVariableFee,
1364     uint256 _variableFeeNumnerator) {
1365     _fixedFee = fixedFee;
1366     _minVariableFee = minVariableFee;
1367     _maxVariableFee = maxVariableFee;
1368     _variableFeeNumnerator = variableFeeNumerator;
1369 }
```

✓ The code meets the specification.

Formal Verification Request 73

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 0.36 ms

Line 1352 in File EURSToken.sol

1352 //CTK NO_ASF

Line 1360-1369 in File EURSToken.sol

```
1360 function getFeeParameters () public /*>IGNORE delegatable IGNORE<*/ view returns (
1361     uint256 _fixedFee,
1362     uint256 _minVariableFee,
1363     uint256 _maxVariableFee,
1364     uint256 _variableFeeNumnerator) {
1365     _fixedFee = fixedFee;
1366     _minVariableFee = minVariableFee;
1367     _maxVariableFee = maxVariableFee;
1368     _variableFeeNumnerator = variableFeeNumerator;
1369 }
```

✓ The code meets the specification.

Formal Verification Request 74

getFeeParameters

📅 18, Jun 2019

🕒 0.38 ms

Line 1353-1359 in File EURSToken.sol

```
1353 /*CTK getFeeParameters
1354     @tag assume_completion
1355     @post _fixedFee == fixedFee
1356     @post _minVariableFee == minVariableFee
1357     @post _maxVariableFee == maxVariableFee
```

```
1358 @post _variableFeeNumnerator == variableFeeNumerator
1359 */
```

Line 1360-1369 in File EURSToken.sol

```
1360 function getFeeParameters () public /*>IGNORE delegatable IGNORE<*/ view returns (
1361     uint256 _fixedFee,
1362     uint256 _minVariableFee,
1363     uint256 _maxVariableFee,
1364     uint256 _variableFeeNumnerator) {
1365     _fixedFee = fixedFee;
1366     _minVariableFee = minVariableFee;
1367     _maxVariableFee = maxVariableFee;
1368     _variableFeeNumnerator = variableFeeNumerator;
1369 }
```

✓ The code meets the specification.

Formal Verification Request 75

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 121.09 ms

Line 1377 in File EURSToken.sol

```
1377 //@CTK FAIL NO_OVERFLOW
```

Line 1388-1396 in File EURSToken.sol

```
1388 function calculateFee (uint256 _amount)
1389     public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 _fee) {
1390     require (_amount <= MAX_TOKENS_COUNT);
1391
1392     _fee = safeMul (_amount, variableFeeNumerator) / FEE_DENOMINATOR;
1393     if (_fee < minVariableFee) _fee = minVariableFee;
1394     if (_fee > maxVariableFee) _fee = maxVariableFee;
1395     _fee = safeAdd (_fee, fixedFee);
1396 }
```

✗ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3     Input = {
4         _amount = 10
5     }
6     This = 0
7     Internal = {
8         __has_assertion_failure = false
9         __has_buf_overflow = false
10        __has_overflow = false
11        __has_returned = false
12        __reverted = false
13        msg = {
14            "gas": 0,
15            "sender": 0,
16            "value": 0
```

```

17     }
18   }
19   Other = {
20     _fee = 0
21     block = {
22       "number": 0,
23       "timestamp": 0
24     }
25   }
26   Address_Map = [
27     {
28       "key": "ALL_OTHERS",
29       "value": {
30         "contract_name": "EURSToken",
31         "balance": 0,
32         "contract": {
33           "FEE_DENOMINATOR": 176,
34           "MAX_FEE_NUMERATOR": 0,
35           "MIN_FEE_NUMERATOR": 0,
36           "MAX_TOKENS_COUNT": 132,
37           "DEFAULT_FEE": 0,
38           "BLACK_LIST_FLAG": 0,
39           "ZERO_FEE_FLAG": 0,
40           "owner": 0,
41           "feeCollector": 0,
42           "tokensCount": 0,
43           "frozen": false,
44           "nonces": [
45             {
46               "key": 0,
47               "value": 0
48             },
49             {
50               "key": "ALL_OTHERS",
51               "value": 16
52             }
53           ],
54           "fixedFee": 242,
55           "minVariableFee": 1,
56           "maxVariableFee": 0,
57           "variableFeeNumerator": 15,
58           "addressFlags": [
59             {
60               "key": "ALL_OTHERS",
61               "value": 0
62             }
63           ],
64           "delegate": 0,
65           "accounts": [
66             {
67               "key": 4,
68               "value": 8
69             },
70             {
71               "key": "ALL_OTHERS",
72               "value": 16
73             }
74           ],

```

```

75     "allowances": [
76         {
77             "key": "ALL_OTHERS",
78             "value": [
79                 {
80                     "key": 0,
81                     "value": 0
82                 },
83                 {
84                     "key": "ALL_OTHERS",
85                     "value": 16
86                 }
87             ]
88         }
89     ],
90     "MAX_UINT256": 240
91 }
92 }
93 }
94 ]
95
96 After Execution:
97     Input = {
98         _amount = 10
99     }
100     This = 0
101     Internal = {
102         __has_assertion_failure = false
103         __has_buf_overflow = false
104         __has_overflow = true
105         __has_returned = false
106         __reverted = false
107         msg = {
108             "gas": 0,
109             "sender": 0,
110             "value": 0
111         }
112     }
113     Other = {
114         _fee = 242
115         block = {
116             "number": 0,
117             "timestamp": 0
118         }
119     }
120     Address_Map = [
121         {
122             "key": "ALL_OTHERS",
123             "value": {
124                 "contract_name": "EURSToken",
125                 "balance": 0,
126                 "contract": {
127                     "FEE_DENOMINATOR": 176,
128                     "MAX_FEE_NUMERATOR": 0,
129                     "MIN_FEE_NUMERATION": 0,
130                     "MAX_TOKENS_COUNT": 132,
131                     "DEFAULT_FEE": 0,
132                     "BLACK_LIST_FLAG": 0,

```


```


133     "ZERO_FEE_FLAG": 0,
134     "owner": 0,
135     "feeCollector": 0,
136     "tokensCount": 0,
137     "frozen": false,
138     "nonces": [
139         {
140             "key": 0,
141             "value": 0
142         },
143         {
144             "key": "ALL_OTHERS",
145             "value": 16
146         }
147     ],
148     "fixedFee": 242,
149     "minVariableFee": 1,
150     "maxVariableFee": 0,
151     "variableFeeNumerator": 15,
152     "addressFlags": [
153         {
154             "key": "ALL_OTHERS",
155             "value": 0
156         }
157     ],
158     "delegate": 0,
159     "accounts": [
160         {
161             "key": 4,
162             "value": 8
163         },
164         {
165             "key": "ALL_OTHERS",
166             "value": 16
167         }
168     ],
169     "allowances": [
170         {
171             "key": "ALL_OTHERS",
172             "value": [
173                 {
174                     "key": 0,
175                     "value": 0
176                 },
177                 {
178                     "key": "ALL_OTHERS",
179                     "value": 16
180                 }
181             ]
182         }
183     ],
184     "MAX_UINT256": 240
185 }
186 }
187 }
188 ]

```

Formal Verification Request 76

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

 18, Jun 2019

 8.36 ms

Line 1378 in File EURSToken.sol

1378 `//@CTK_NO_BUF_OVERFLOW`

Line 1388-1396 in File EURSToken.sol


```
1388 function calculateFee (uint256 _amount)
1389     public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 _fee) {
1390     require (_amount <= MAX_TOKENS_COUNT);
1391
1392     _fee = safeMul (_amount, variableFeeNumerator) / FEE_DENOMINATOR;
1393     if (_fee < minVariableFee) _fee = minVariableFee;
1394     if (_fee > maxVariableFee) _fee = maxVariableFee;
1395     _fee = safeAdd (_fee, fixedFee);
1396 }
```

 The code meets the specification.

Formal Verification Request 77

Method will not encounter an assertion failure.

 18, Jun 2019

 38.54 ms

Line 1379 in File EURSToken.sol

1379 `//@CTK_FAIL_NO_ASF`

Line 1388-1396 in File EURSToken.sol

```
1388 function calculateFee (uint256 _amount)
1389     public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 _fee) {
1390     require (_amount <= MAX_TOKENS_COUNT);
1391
1392     _fee = safeMul (_amount, variableFeeNumerator) / FEE_DENOMINATOR;
1393     if (_fee < minVariableFee) _fee = minVariableFee;
1394     if (_fee > maxVariableFee) _fee = maxVariableFee;
1395     _fee = safeAdd (_fee, fixedFee);
1396 }
```

 This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3     Input = {
4         _amount = 0
5     }
6     This = 0
7     Internal = {
8         __has_assertion_failure = false
9         __has_buf_overflow = false
```

```

10     __has_overflow = false
11     __has_returned = false
12     __reverted = false
13     msg = {
14         "gas": 0,
15         "sender": 0,
16         "value": 0
17     }
18 }
19 Other = {
20     _fee = 0
21     block = {
22         "number": 0,
23         "timestamp": 0
24     }
25 }
26 Address_Map = [
27     {
28         "key": "ALL_OTHERS",
29         "value": {
30             "contract_name": "EURSToken",
31             "balance": 0,
32             "contract": {
33                 "FEE_DENOMINATOR": 14,
34                 "MAX_FEE_NUMERATOR": 0,
35                 "MIN_FEE_NUMERATION": 0,
36                 "MAX_TOKENS_COUNT": 0,
37                 "DEFAULT_FEE": 0,
38                 "BLACK_LIST_FLAG": 0,
39                 "ZERO_FEE_FLAG": 0,
40                 "owner": 0,
41                 "feeCollector": 0,
42                 "tokensCount": 0,
43                 "frozen": false,
44                 "nonces": [
45                     {
46                         "key": "ALL_OTHERS",
47                         "value": 0
48                     }
49                 ],
50                 "fixedFee": 0,
51                 "minVariableFee": 32,
52                 "maxVariableFee": 128,
53                 "variableFeeNumerator": 5,
54                 "addressFlags": [
55                     {
56                         "key": 0,
57                         "value": 2
58                     },
59                     {
60                         "key": "ALL_OTHERS",
61                         "value": 64
62                     }
63                 ],
64                 "delegate": 0,
65                 "accounts": [
66                     {
67                         "key": 0,

```



```


68         "value": 0
69     },
70     {
71         "key": "ALL_OTHERS",
72         "value": 4
73     }
74 ],
75 "allowances": [
76     {
77         "key": "ALL_OTHERS",
78         "value": [
79             {
80                 "key": "ALL_OTHERS",
81                 "value": 0
82             }
83         ]
84     }
85 ],
86 "MAX_UINT256": 0
87 }
88 }
89 }
90 ]
91
92 Function invocation is reverted.

```

Formal Verification Request 78

calculateFee

 18, Jun 2019

 2703.68 ms

Line 1380-1387 in File EURSToken.sol

```

1380  /*@CTK calculateFee
1381      @tag assume_completion
1382      @pre _amount <= MAX_TOKENS_COUNT
1383      @pre maxVariableFee > minVariableFee
1384      @post (_amount * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee -> _fee
1385          == fixedFee + maxVariableFee
1386      @post (_amount * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee -> _fee
1387          == fixedFee + minVariableFee
1388      @post ((_amount * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee && (
1389          _amount * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee) -> _fee
1390          == fixedFee + (_amount * variableFeeNumerator) / FEE_DENOMINATOR
1391  */

```

Line 1388-1396 in File EURSToken.sol

```

1388  function calculateFee (uint256 _amount)
1389      public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 _fee) {
1390      require (_amount <= MAX_TOKENS_COUNT);
1391
1392      _fee = safeMul (_amount, variableFeeNumerator) / FEE_DENOMINATOR;
1393      if (_fee < minVariableFee) _fee = minVariableFee;
1394      if (_fee > maxVariableFee) _fee = maxVariableFee;
1395      _fee = safeAdd (_fee, fixedFee);


```


1396 }

✓ The code meets the specification.

Formal Verification Request 79

If method completes, integer overflow would not happen.

 18, Jun 2019

 15.43 ms

Line 1404 in File EURSToken.sol

1404 // @CTK NO_OVERFLOW

Line 1412-1417 in File EURSToken.sol


```
1412 function setFlags (address _address, uint256 _flags)
1413 public /*>IGNORE delegatable IGNORE<*/ payable {
1414     require (msg.sender == owner);
1415
1416     addressFlags [_address] = _flags;
1417 }
```

✓ The code meets the specification.

Formal Verification Request 80

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

 18, Jun 2019

 0.45 ms

Line 1405 in File EURSToken.sol

1405 // @CTK NO_BUF_OVERFLOW

Line 1412-1417 in File EURSToken.sol


```
1412 function setFlags (address _address, uint256 _flags)
1413 public /*>IGNORE delegatable IGNORE<*/ payable {
1414     require (msg.sender == owner);
1415
1416     addressFlags [_address] = _flags;
1417 }
```

✓ The code meets the specification.

Formal Verification Request 81

Method will not encounter an assertion failure.

 18, Jun 2019

 0.4 ms

Line 1406 in File EURSToken.sol

1406 // @CTK NO_ASF


Line 1412-1417 in File EURSToken.sol


```
1412 function setFlags (address _address, uint256 _flags)
1413 public /*>IGNORE delegatable IGNORE<*/ payable {
1414     require (msg.sender == owner);
1415
1416     addressFlags [_address] = _flags;
1417 }
```

✓ The code meets the specification.

Formal Verification Request 82

setFlags

 18, Jun 2019

 2.85 ms

Line 1407-1411 in File EURSToken.sol

```
1407 /*@CTK setFlags
1408    @tag assume_completion
1409    @pre msg.sender == owner
1410    @post __post.addressFlags [_address] == _flags
1411 */
```

Line 1412-1417 in File EURSToken.sol


```
1412 function setFlags (address _address, uint256 _flags)
1413 public /*>IGNORE delegatable IGNORE<*/ payable {
1414     require (msg.sender == owner);
1415
1416     addressFlags [_address] = _flags;
1417 }
```

✓ The code meets the specification.

Formal Verification Request 83

If method completes, integer overflow would not happen.

 18, Jun 2019

 19.24 ms

Line 1437 in File EURSToken.sol

1437 // @CTK NO_OVERFLOW

Line 1445-1452 in File EURSToken.sol

```
1445 function setDelegate (address _delegate) public {
1446     require (msg.sender == owner);
1447
1448     if (delegate != _delegate) {
1449         delegate = _delegate;
1450         Delegation (delegate);
1451     }
1452 }
```


```
1451   }  
1452   }
```

✓ The code meets the specification.

Formal Verification Request 84

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

 18, Jun 2019

 0.47 ms

Line 1438 in File EURSToken.sol

```
1438   //@CTK NO_BUF_OVERFLOW
```

Line 1445-1452 in File EURSToken.sol


```
1445   function setDelegate (address _delegate) public {  
1446       require (msg.sender == owner);  
1447  
1448       if (delegate != _delegate) {  
1449           delegate = _delegate;  
1450           Delegation (delegate);  
1451       }  
1452   }
```

✓ The code meets the specification.

Formal Verification Request 85

Method will not encounter an assertion failure.

 18, Jun 2019

 0.45 ms

Line 1439 in File EURSToken.sol

```
1439   //@CTK NO_ASF
```


Line 1445-1452 in File EURSToken.sol


```
1445   function setDelegate (address _delegate) public {  
1446       require (msg.sender == owner);  
1447  
1448       if (delegate != _delegate) {  
1449           delegate = _delegate;  
1450           Delegation (delegate);  
1451       }  
1452   }
```

✓ The code meets the specification.

Formal Verification Request 86

setDelegate

 18, Jun 2019

 6.14 ms

Line 1440-1444 in File EURSToken.sol

```
1440  /*@CTK "setDelegate"
1441     @tag assume_completion
1442     @pre msg.sender == owner
1443     @post __post.delegate == _delegate
1444  */
```

Line 1445-1452 in File EURSToken.sol

```
1445  function setDelegate (address _delegate) public {
1446      require (msg.sender == owner);
1447
1448      if (delegate != _delegate) {
1449          delegate = _delegate;
1450          Delegation (delegate);
1451      }
1452  }
```

 The code meets the specification.

Source Code with CertiK Labels

File EURSToken.sol

```

1  /**
2   *Submitted for verification at Etherscan.io on 2018-07-03
3   */
4
5  /**
6   * EURS Token Smart Contract: EIP-20 compatible token smart contract that
7   * manages EURS tokens.
8   */
9
10 pragma solidity ^0.4.20;
11
12 /**
13  * Provides methods to safely add, subtract and multiply uint256 numbers.
14  */
15 contract SafeMath {
16     uint256 constant private MAX_UINT256 =
17         0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF;
18
19     /**
20      * Add two uint256 values, throw in case of overflow.
21      *
22      * @param x first value to add
23      * @param y second value to add
24      * @return x + y
25      */
26     //@CTK FAIL NO_ASF
27     /*@CTK "SafeMath add"
28      @tag spec
29      @pre (MAX_UINT256) == (0xFF)
30      @post (x > (MAX_UINT256 - y)) == __reverted
31      @post (!__reverted) -> z == (x + y)
32      @post (!__reverted) -> !__has_overflow
33      @post !(__has_buf_overflow)
34     */
35     function safeAdd (uint256 x, uint256 y)
36     pure internal
37     returns (uint256 z) {
38         assert (x <= MAX_UINT256 - y);
39         return x + y;
40     }
41
42     /**
43      * Subtract one uint256 value from another, throw in case of underflow.
44      *
45      * @param x value to subtract from
46      * @param y value to subtract
47      * @return x - y
48      */
49     //@CTK FAIL NO_ASF
50     /*@CTK "SafeMath sub"
51      @tag spec
52      @post (x < y) == (__reverted)
53      @post (!__reverted) -> (z == (x - y))
54      @post (!__reverted) -> (!__has_overflow)

```

```

55     @post !(__has_buf_overflow)
56 */
57 function safeSub (uint256 x, uint256 y)
58 pure internal
59 returns (uint256 z) {
60     assert (x >= y);
61     return x - y;
62 }
63
64 /**
65  * Multiply two uint256 values, throw in case of overflow.
66  *
67  * @param x first value to multiply
68  * @param y second value to multiply
69  * @return x * y
70  */
71 //@CTK FAIL NO_ASF
72 /*@CTK "SafeMath mul"
73  @tag spec
74  @pre (MAX_UINT256) == (0xFF)
75  @post (y == 0) -> (z == 0)
76  @post (x > (MAX_UINT256 / y)) == (__reverted)
77  @post (!__reverted) -> (z == (x * y))
78  @post (!__reverted) -> (!__has_overflow)
79  @post !(__has_buf_overflow)
80 */
81 function safeMul (uint256 x, uint256 y)
82 pure internal
83 returns (uint256 z) {
84     if (y == 0) return 0; // Prevent division by zero at the next line
85     assert (x <= MAX_UINT256 / y);
86     return x * y;
87 }
88 }
89 /*
90  * EIP-20 Standard Token Smart Contract Interface.
91  * Copyright (c) 2018 by STSS (Malta) Limited.
92  * Contact: <tech@stasis.net>
93
94  * ERC-20 standard token interface, as defined
95  * <a href="https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md">here</a>.
96  */
97 contract Token {
98     /**
99     * Get total number of tokens in circulation.
100     *
101     * @return total number of tokens in circulation
102     */
103     function totalSupply () public view returns (uint256 supply);
104
105     /**
106     * Get number of tokens currently belonging to given owner.
107     *
108     * @param _owner address to get number of tokens currently belonging to the
109     *         owner of
110     * @return number of tokens currently belonging to the owner of given address
111     */
112     function balanceOf (address _owner) public view returns (uint256 balance);

```

```

113
114 /**
115  * Transfer given number of tokens from message sender to given recipient.
116  *
117  * @param _to address to transfer tokens to the owner of
118  * @param _value number of tokens to transfer to the owner of given address
119  * @return true if tokens were transferred successfully, false otherwise
120  */
121 function transfer (address _to, uint256 _value)
122 public payable returns (bool success);
123
124 /**
125  * Transfer given number of tokens from given owner to given recipient.
126  *
127  * @param _from address to transfer tokens from the owner of
128  * @param _to address to transfer tokens to the owner of
129  * @param _value number of tokens to transfer from given owner to given
130  *       recipient
131  * @return true if tokens were transferred successfully, false otherwise
132  */
133 function transferFrom (address _from, address _to, uint256 _value)
134 public payable returns (bool success);
135
136 /**
137  * Allow given spender to transfer given number of tokens from message sender.
138  *
139  * @param _spender address to allow the owner of to transfer tokens from
140  *       message sender
141  * @param _value number of tokens to allow to transfer
142  * @return true if token transfer was successfully approved, false otherwise
143  */
144 function approve (address _spender, uint256 _value)
145 public payable returns (bool success);
146
147 /**
148  * Tell how many tokens given spender is currently allowed to transfer from
149  * given owner.
150  *
151  * @param _owner address to get number of tokens allowed to be transferred
152  *       from the owner of
153  * @param _spender address to get number of tokens allowed to be transferred
154  *       by the owner of
155  * @return number of tokens given spender is currently allowed to transfer
156  *       from given owner
157  */
158 function allowance (address _owner, address _spender)
159 public view returns (uint256 remaining);
160
161 /**
162  * Logged when tokens were transferred from one owner to another.
163  *
164  * @param _from address of the owner, tokens were transferred from
165  * @param _to address of the owner, tokens were transferred to
166  * @param _value number of tokens transferred
167  */
168 event Transfer (address indexed _from, address indexed _to, uint256 _value);
169
170 /**

```



```

171  * Logged when owner approved his tokens to be transferred by some spender.
172  *
173  * @param _owner owner who approved his tokens to be transferred
174  * @param _spender spender who were allowed to transfer the tokens belonging
175  *       to the owner
176  * @param _value number of tokens belonging to the owner, approved to be
177  *       transferred by the spender
178  */
179  event Approval (
180      address indexed _owner, address indexed _spender, uint256 _value);
181  }
182  /*
183  * Abstract Token Smart Contract.
184  * Copyright (c) 2018 by STSS (Malta) Limited.
185  * Contact: <tech@stasis.net>
186
187  * Abstract Token Smart Contract that could be used as a base contract for
188  * ERC-20 token contracts.
189  */
190  contract AbstractToken is Token, SafeMath {
191      /**
192       * Create new Abstract Token contract.
193       */
194      function AbstractToken () public {
195          // Do nothing
196      }
197
198      /**
199       * Get number of tokens currently belonging to given owner.
200       *
201       * @param _owner address to get number of tokens currently belonging to the
202       *       owner of
203       * @return number of tokens currently belonging to the owner of given address
204       */
205      /*@CTK "balanceOf"
206       @tag spec
207       @tag assume_completion
208       @post balance == accounts [_owner]
209       */
210      function balanceOf (address _owner) public view returns (uint256 balance) {
211          return accounts [_owner];
212      }
213
214      /**
215       * Transfer given number of tokens from message sender to given recipient.
216       *
217       * @param _to address to transfer tokens to the owner of
218       * @param _value number of tokens to transfer to the owner of given address
219       * @return true if tokens were transferred successfully, false otherwise
220       */
221      /*@CTK FAIL NO_OVERFLOW
222       /*@CTK NO_BUF_OVERFLOW
223       /*@CTK FAIL NO_ASF
224       /*@CTK "transfer error"
225       @tag spec
226       @tag assume_completion
227       @post (accounts[msg.sender] < _value) -> (success == false)

```

```

228     @post (accounts[msg.sender] >= _value && msg.sender != _to) -> (__post.accounts[
        msg.sender] == accounts[msg.sender] - _value) && (__post.accounts[_to] ==
        accounts[_to] + _value)
229     @post (accounts[msg.sender] < _value || _value == 0 || msg.sender == _to) -> (
        __post.accounts[msg.sender] == accounts[msg.sender]) && (__post.accounts[_to]
        == accounts[_to])
230 */
231 /*@CTK "transfer"
232 @tag spec
233 @tag assume_completion
234 @post (accounts[msg.sender] < _value) -> (success == false)
235 @post (accounts[msg.sender] >= _value && msg.sender != _to) -> (__post.accounts[
        msg.sender] == accounts[msg.sender] - _value) && (__post.accounts[_to] ==
        accounts[_to] + _value)
236 @post (accounts[msg.sender] < _value || _value == 0 || msg.sender == _to) -> (
        __post.accounts[msg.sender] == accounts[msg.sender]) && (__post.accounts[_to]
        == accounts[_to])
237 */
238 function transfer (address _to, uint256 _value)
239 public payable returns (bool success) {
240     uint256 fromBalance = accounts [msg.sender];
241     if (fromBalance < _value) return false;
242     if (_value > 0 && msg.sender != _to) {
243         accounts [msg.sender] = safeSub (fromBalance, _value);
244         accounts [_to] = safeAdd (accounts [_to], _value);
245     }
246     Transfer (msg.sender, _to, _value);
247     return true;
248 }
249
250 /**
251  * Transfer given number of tokens from given owner to given recipient.
252  *
253  * @param _from address to transfer tokens from the owner of
254  * @param _to address to transfer tokens to the owner of
255  * @param _value number of tokens to transfer from given owner to given
256  *         recipient
257  * @return true if tokens were transferred successfully, false otherwise
258  */
259 // @CTK FAIL NO_OVERFLOW
260 // @CTK NO_BUF_OVERFLOW
261 // @CTK FAIL NO_ASF
262 /*@CTK "transferFrom"
263 @tag spec
264 @tag assume_completion
265 @post (allowances[_from][msg.sender] < _value) -> (success == false)
266 @post (accounts [_from] < _value) -> (success == false)
267 @post (allowances[_from][msg.sender] >= _value && accounts [_from] >= _value) -> (
        __post.allowances[_from][msg.sender] == allowances[_from][msg.sender] - _value
        )
268 @post (allowances[_from][msg.sender] < _value || accounts [_from] < _value) -> (
        __post.allowances[_from][msg.sender] == allowances[_from][msg.sender])
269 @post (allowances[_from][msg.sender] >= _value && accounts [_from] >= _value &&
        _from != _to) -> (__post.accounts[_from] == accounts [_from] - _value) && (
        __post.accounts[_to] == accounts [_to] + _value)
270 @post (allowances[_from][msg.sender] < _value || accounts [_from] < _value ||
        _from == _to) -> (__post.accounts[_from] == accounts [_from]) && (__post.
        accounts[_to] == accounts [_to])

```

```

271 */
272 function transferFrom (address _from, address _to, uint256 _value)
273 public payable returns (bool success) {
274     uint256 spenderAllowance = allowances [_from][msg.sender];
275     if (spenderAllowance < _value) return false;
276     uint256 fromBalance = accounts [_from];
277     if (fromBalance < _value) return false;
278
279     allowances [_from][msg.sender] =
280         safeSub (spenderAllowance, _value);
281
282     if (_value > 0 && _from != _to) {
283         accounts [_from] = safeSub (fromBalance, _value);
284         accounts [_to] = safeAdd (accounts [_to], _value);
285     }
286     Transfer (_from, _to, _value);
287     return true;
288 }
289
290 /**
291  * Allow given spender to transfer given number of tokens from message sender.
292  *
293  * @param _spender address to allow the owner of to transfer tokens from
294  *         message sender
295  * @param _value number of tokens to allow to transfer
296  * @return true if token transfer was successfully approved, false otherwise
297  */
298 // @CTK NO_OVERFLOW
299 // @CTK NO_BUF_OVERFLOW
300 // @CTK NO_ASF
301 /* @CTK "approve"
302    @tag spec
303    @tag assume_completion
304    @post (__post.allowances[msg.sender][_spender]) == (_value)
305 */
306 function approve (address _spender, uint256 _value)
307 public payable returns (bool success) {
308     allowances [msg.sender][_spender] = _value;
309     Approval (msg.sender, _spender, _value);
310
311     return true;
312 }
313
314 /**
315  * Tell how many tokens given spender is currently allowed to transfer from
316  * given owner.
317  *
318  * @param _owner address to get number of tokens allowed to be transferred
319  *         from the owner of
320  * @param _spender address to get number of tokens allowed to be transferred
321  *         by the owner of
322  * @return number of tokens given spender is currently allowed to transfer
323  *         from given owner
324  */
325 // @CTK NO_OVERFLOW
326 // @CTK NO_BUF_OVERFLOW
327 // @CTK NO_ASF
328 /* @CTK "allowance"

```

```

329     @tag spec
330     @tag assume_completion
331     @post (remaining) == (allowances[_owner][_spender])
332 */
333 function allowance (address _owner, address _spender)
334 public view returns (uint256 remaining) {
335     return allowances [_owner][_spender];
336 }
337
338 /**
339  * Mapping from addresses of token holders to the numbers of tokens belonging
340  * to these token holders.
341  */
342 mapping (address => uint256) internal accounts;
343
344 /**
345  * Mapping from addresses of token holders to the mapping of addresses of
346  * spenders to the allowances set by these token holders to these spenders.
347  */
348 mapping (address => mapping (address => uint256)) internal allowances;
349 }
350
351 /*
352  * EURS Token Smart Contract.
353  * Copyright (c) 2018 by STSS (Malta) Limited.
354  * Contact: <tech@stasis.net>
355  */
356
357 contract EURSToken is AbstractToken {
358     /**
359      * Fee denominator (0.001%).
360      */
361     uint256 constant internal FEE_DENOMINATOR = 100000;
362
363     /**
364      * Maximum fee numerator (100%).
365      */
366     uint256 constant internal MAX_FEE_NUMERATOR = FEE_DENOMINATOR;
367
368     /**
369      * Minimum fee numerator (0%).
370      */
371     uint256 constant internal MIN_FEE_NUMERATOR = 0;
372
373     /**
374      * Maximum allowed number of tokens in circulation.
375      */
376     uint256 constant internal MAX_TOKENS_COUNT =
377         0xffffffffffffffffffffffffffffffffffffffffffffffffffffffff /
378         MAX_FEE_NUMERATOR;
379
380     /**
381      * Default transfer fee.
382      */
383     uint256 constant internal DEFAULT_FEE = 5e2;
384
385     /**
386      * Address flag that marks black listed addresses.

```

```

387  */
388  uint256 constant internal BLACK_LIST_FLAG = 0x01;
389
390  /**
391   * Address flag that marks zero fee addresses.
392   */
393  uint256 constant internal ZERO_FEE_FLAG = 0x02;
394
395  modifier delegatable {
396      if (delegate == address (0)) {
397          require (msg.value == 0); // Non payable if not delegated
398          _;
399      } else {
400          //@CTK NO_OVERFLOW
401          //@CTK NO_BUF_OVERFLOW
402          //@CTK NO_ASF
403          /*@CTK "assembly call status"
404             @post !(__reverted) -> delegate == delegate__pre
405             @post !(__reverted) -> owner == owner__pre
406          */
407          assembly {
408              // Save owner
409              let oldOwner := sload (owner_slot)
410
411              // Save delegate
412              let oldDelegate := sload (delegate_slot)
413
414              // Solidity stores address of the beginning of free memory at 0x40
415              let buffer := mload (0x40)
416
417              // Copy message call data into buffer
418              calldatacopy (buffer, 0, calldatasize)
419
420              // Lets call our delegate
421              let result := delegatecall (gas, oldDelegate, buffer, calldatasize, buffer, 0)
422
423              // Check, whether owner was changed
424              switch eq (oldOwner, sload (owner_slot))
425              case 1 {} // Owner was not changed, fine
426              default {revert (0, 0)} // Owner was changed, revert!
427
428              // Check, whether delegate was changed
429              switch eq (oldDelegate, sload (delegate_slot))
430              case 1 {} // Delegate was not changed, fine
431              default {revert (0, 0)} // Delegate was changed, revert!
432
433              // Copy returned value into buffer
434              returndatacopy (buffer, 0, returndatasize)
435
436              // Check call status
437              switch result
438              case 0 { revert (buffer, returndatasize) } // Call failed, revert!
439              default { return (buffer, returndatasize) } // Call succeeded, return
440          }
441      }
442  }
443
444  /**

```

```

445  * Create EURS Token smart contract with message sender as an owner.
446  *
447  * @param _feeCollector address fees are sent to
448  */
449  //@CTK NO_OVERFLOW
450  //@CTK NO_BUF_OVERFLOW
451  //@CTK NO_ASF
452  /*@CTK "EURSToken"
453   @tag assume_completion
454   @post __post.fixedFee == DEFAULT_FEE
455   @post __post.minVariableFee == 0
456   @post __post.maxVariableFee == 0
457   @post __post.variableFeeNumerator == 0
458   @post __post.owner == msg.sender
459   @post __post.feeCollector == _feeCollector
460  */
461  function EURSToken (address _feeCollector) public {
462      fixedFee = DEFAULT_FEE;
463      minVariableFee = 0;
464      maxVariableFee = 0;
465      variableFeeNumerator = 0;
466
467      owner = msg.sender;
468      feeCollector = _feeCollector;
469  }
470
471  /**
472   * Delegate unrecognized functions.
473   */
474  function () public payable {
475      revert (); // Revert if not delegated
476  }
477
478  /**
479   * Get name of the token.
480   *
481   * @return name of the token
482   */
483  function name () public delegatable view returns (string) {
484      return "STASIS EURS Token";
485  }
486
487  /**
488   * Get symbol of the token.
489   *
490   * @return symbol of the token
491   */
492  function symbol () public delegatable view returns (string) {
493      return "EURS";
494  }
495
496  /**
497   * Get number of decimals for the token.
498   *
499   * @return number of decimals for the token
500   */
501  function decimals () public delegatable view returns (uint8) {
502      return 2;

```

```

503 }
504
505 /**
506  * Get total number of tokens in circulation.
507  *
508  * @return total number of tokens in circulation
509  */
510 /*@CTK "EURSToken totalSupply"
511  @tag assume_completion
512  @post __return == tokensCount
513  */
514 function totalSupply () public /*>IGNORE delegatable IGNORE<*/ view returns (uint256
515 ) {
516     return tokensCount;
517 }
518
519 /**
520  * Get number of tokens currently belonging to given owner.
521  *
522  * @param _owner address to get number of tokens currently belonging to the
523  *         owner of
524  * @return number of tokens currently belonging to the owner of given address
525  */
526 /*@CTK "EURSToken balanceOf"
527  @tag assume_completion
528  @post balance == accounts[_owner]
529  */
530 function balanceOf (address _owner)
531     public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 balance) {
532     return AbstractToken.balanceOf (_owner);
533 }
534
535 /**
536  * Transfer given number of tokens from message sender to given recipient.
537  *
538  * @param _to address to transfer tokens to the owner of
539  * @param _value number of tokens to transfer to the owner of given address
540  * @return true if tokens were transferred successfully, false otherwise
541  */
542 /*@CTK "transfer frozen"
543  @tag spec
544  @tag assume_completion
545  @pre frozen
546  @post (__post.accounts[msg.sender] == accounts[msg.sender])
547  @post (__post.accounts[_to] == accounts[_to])
548  @post (__post.accounts[feeCollector] == accounts[feeCollector])
549  @post !__return
550  */
551 /*@CTK "transfer blacklisted"
552  @tag spec
553  @tag assume_completion
554  @pre !frozen
555  @pre (addressFlags[msg.sender] == BLACK_LIST_FLAG) || (addressFlags [_to] ==
556         BLACK_LIST_FLAG)
557  @post (__post.accounts[msg.sender] == accounts[msg.sender])
558  @post (__post.accounts[_to] == accounts[_to])
559  @post (__post.accounts[feeCollector] == accounts[feeCollector])
560  @post !__return

```

```

559  */
560  /*@CTK "transfer zero-feed succeed"
561    @tag spec
562    @tag assume_completion
563    @pre !frozen
564    @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
565    @pre (addressFlags[msg.sender] == ZERO_FEE_FLAG) || (addressFlags [_to] ==
        ZERO_FEE_FLAG)
566    @pre (_value <= accounts[msg.sender])
567    @post (__post.accounts[msg.sender] == accounts[msg.sender] - _value)
568    @post (__post.accounts[_to] == accounts[_to] + _value)
569    @post (__post.accounts[feeCollector] == accounts[feeCollector])
570    @post __return
571  */
572  /*@CTK "transfer zero-feed fail"
573    @tag spec
574    @tag assume_completion
575    @pre !frozen
576    @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
577    @pre (addressFlags[msg.sender] == ZERO_FEE_FLAG) || (addressFlags [_to] ==
        ZERO_FEE_FLAG)
578    @pre (_value > accounts[msg.sender])
579    @post (__post.accounts[msg.sender] == accounts[msg.sender])
580    @post (__post.accounts[_to] == accounts[_to])
581    @post (__post.accounts[feeCollector] == accounts[feeCollector])
582    @post __return
583  */
584  /*@CTK "transfer maxVariableFee succeed"
585    @tag spec
586    @tag assume_completion
587    @pre !frozen
588    @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
589    @pre (addressFlags[msg.sender] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
        ZERO_FEE_FLAG)
590    @pre maxVariableFee > minVariableFee
591    @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee
592    @pre (_value + fixedFee + maxVariableFee) <= accounts[msg.sender]
593    @post __post.accounts[msg.sender] == accounts[msg.sender] - (_value + fixedFee +
        maxVariableFee)
594    @post __post.accounts[_to] == accounts[_to] + _value
595    @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee +
        maxVariableFee)
596    @post __return
597  */
598  /*#CTK "transfer maxVariableFee fail"
599    @tag spec
600    @tag assume_completion
601    @pre !frozen
602    @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
603    @pre (addressFlags[msg.sender] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
        ZERO_FEE_FLAG)
604    @pre maxVariableFee > minVariableFee
605    @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee
606    @pre (_value + fixedFee + maxVariableFee) > accounts[msg.sender]

```



```

607     @post __post.accounts[msg.sender] == accounts[msg.sender]
608     @post __post.accounts[_to] == accounts[_to]
609     @post __post.accounts[feeCollector] == accounts[feeCollector]
610     @post !__return
611 */
612 /*@CTK "transfer minVariableFee succeed"
613     @tag spec
614     @tag assume_completion
615     @pre !frozen
616     @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
617     @pre (addressFlags[msg.sender] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
        ZERO_FEE_FLAG)
618     @pre maxVariableFee > minVariableFee
619     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee
620     @pre (_value + fixedFee + minVariableFee) <= accounts[msg.sender]
621     @post __post.accounts[msg.sender] == accounts[msg.sender] - (_value + fixedFee +
        minVariableFee)
622     @post __post.accounts[_to] == accounts[_to] + _value
623     @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee +
        minVariableFee)
624     @post __return
625 */
626 /*@CTK "transfer minVariableFee fail"
627     @tag spec
628     @tag assume_completion
629     @pre !frozen
630     @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
631     @pre (addressFlags[msg.sender] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
        ZERO_FEE_FLAG)
632     @pre maxVariableFee > minVariableFee
633     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee
634     @pre (_value + fixedFee + minVariableFee) > accounts[msg.sender]
635     @post __post.accounts[msg.sender] == accounts[msg.sender]
636     @post __post.accounts[_to] == accounts[_to]
637     @post __post.accounts[feeCollector] == accounts[feeCollector]
638     @post !__return
639 */
640 /*@CTK "transfer normal variableFee succeed"
641     @tag spec
642     @tag assume_completion
643     @pre !frozen
644     @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
645     @pre (addressFlags[msg.sender] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
        ZERO_FEE_FLAG)
646     @pre maxVariableFee > minVariableFee
647     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee
648     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee
649     @pre (_value + fixedFee + (_value * variableFeeNumerator) / FEE_DENOMINATOR) <=
        accounts[msg.sender]
650     @post __post.accounts[msg.sender] == accounts[msg.sender] - (_value + fixedFee + (
        _value * variableFeeNumerator) / FEE_DENOMINATOR)
651     @post __post.accounts[_to] == accounts[_to] + _value
652     @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee + (
        _value * variableFeeNumerator) / FEE_DENOMINATOR)
653     @post __return

```

```

654  */
655  /*@CTK "transfer normal variableFee fail"
656    @tag spec
657    @tag assume_completion
658    @pre !frozen
659    @pre (addressFlags[msg.sender] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
        BLACK_LIST_FLAG)
660    @pre (addressFlags[msg.sender] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
        ZERO_FEE_FLAG)
661    @pre maxVariableFee > minVariableFee
662    @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee
663    @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee
664    @pre (_value + fixedFee + (_value * variableFeeNumerator) / FEE_DENOMINATOR) >
        accounts[msg.sender]
665    @post __post.accounts[msg.sender] == accounts[msg.sender]
666    @post __post.accounts[_to] == accounts[_to]
667    @post __post.accounts[feeCollector] == accounts[feeCollector]
668    @post !__return
669  */
670  function transfer (address _to, uint256 _value)
671  public delegatable payable returns (bool) {
672    if (frozen) return false;
673    else if (
674      (addressFlags [msg.sender] | addressFlags [_to]) & BLACK_LIST_FLAG ==
675      BLACK_LIST_FLAG)
676      return false;
677    else {
678      uint256 fee =
679        (addressFlags [msg.sender] | addressFlags [_to]) & ZERO_FEE_FLAG ==
680        ZERO_FEE_FLAG ?
681        0 :
682        calculateFee (_value);
683      if (_value <= accounts [msg.sender] &&
684        fee <= safeSub (accounts [msg.sender], _value)) {
685        require (AbstractToken.transfer (_to, _value));
686        require (AbstractToken.transfer (feeCollector, fee));
687        return true;
688      } else return false;
689    }
690  }
691
692  /**
693   * Transfer given number of tokens from given owner to given recipient.
694   *
695   * @param _from address to transfer tokens from the owner of
696   * @param _to address to transfer tokens to the owner of
697   * @param _value number of tokens to transfer from given owner to given
698   *       recipient
699   * @return true if tokens were transferred successfully, false otherwise
700   */
701  /*@CTK "transferFrom frozen"
702    @tag spec
703    @tag assume_completion
704    @pre frozen
705    @post (__post.accounts[_from] == accounts[_from])
706    @post (__post.accounts[_to] == accounts[_to])
707    @post (__post.accounts[feeCollector] == accounts[feeCollector])

```

```

708     @post (__post.allowances[_from][msg.sender] == allowances[_from][msg.sender])
709     @post !__return
710 */
711 /*@CTK "transferFrom blacklisted"
712     @tag spec
713     @tag assume_completion
714     @pre !frozen
715     @pre (addressFlags[_from] == BLACK_LIST_FLAG) || (addressFlags[_to] ==
        BLACK_LIST_FLAG)
716     @post (__post.accounts[_from] == accounts[_from])
717     @post (__post.accounts[_to] == accounts[_to])
718     @post (__post.accounts[feeCollector] == accounts[feeCollector])
719     @post (__post.allowances[_from][msg.sender] == allowances[_from][msg.sender])
720     @post !__return
721 */
722 /*@CTK "transferFrom zero-feed succeed"
723     @tag spec
724     @tag assume_completion
725     @pre !frozen
726     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags[_to] !=
        BLACK_LIST_FLAG)
727     @pre (addressFlags[_from] == ZERO_FEE_FLAG) || (addressFlags[_to] ==
        ZERO_FEE_FLAG)
728     @pre (_value <= accounts[_from]) && (_value <= allowances[_from][msg.sender])
729     @post (__post.accounts[_from] == accounts[_from] - _value)
730     @post (__post.accounts[_to] == accounts[_to] + _value)
731     @post (__post.accounts[feeCollector] == accounts[feeCollector])
732     @post (__post.allowances[_from][msg.sender] == allowances[_from][msg.sender] -
        _value)
733     @post __return
734 */
735 /*@CTK "transferFrom zero-feed fail"
736     @tag spec
737     @tag assume_completion
738     @pre !frozen
739     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags[_to] !=
        BLACK_LIST_FLAG)
740     @pre (addressFlags[_from] == ZERO_FEE_FLAG) || (addressFlags[_to] ==
        ZERO_FEE_FLAG)
741     @pre (_value > accounts[_from]) || (_value > allowances[_from][msg.sender])
742     @post (__post.accounts[_from] == accounts[_from])
743     @post (__post.accounts[_to] == accounts[_to])
744     @post (__post.accounts[feeCollector] == accounts[feeCollector])
745     @post (__post.allowances[_from][msg.sender] == allowances[_from][msg.sender])
746     @post __return
747 */
748 /*@CTK "transferFrom maxVariableFee succeed"
749     @tag spec
750     @tag assume_completion
751     @pre !frozen
752     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags[_to] !=
        BLACK_LIST_FLAG)
753     @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags[_to] !=
        ZERO_FEE_FLAG)
754     @pre maxVariableFee > minVariableFee
755     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee
756     @pre (_value + fixedFee + maxVariableFee) <= accounts[_from] && (_value + fixedFee
        + maxVariableFee) <= allowances[_from][msg.sender]

```

```

757     @post __post.accounts[_from] == accounts[_from] - (_value + fixedFee +
758         maxVariableFee)
759     @post __post.accounts[_to] == accounts[_to] + _value
760     @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee +
761         maxVariableFee)
762     @post __post.allowances[_from][msg.sender] == allowances[_from][msg.sender] - (
763         _value + fixedFee + maxVariableFee)
764     @post __return
765 */
766 /*@CTK "transferFrom maxVariableFee fail"
767     @tag spec
768     @tag assume_completion
769     @pre !frozen
770     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
771         BLACK_LIST_FLAG)
772     @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
773         ZERO_FEE_FLAG)
774     @pre maxVariableFee > minVariableFee
775     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee
776     @pre (_value + fixedFee + maxVariableFee) > accounts[_from] || (_value + fixedFee
777         + maxVariableFee) > allowances [_from][msg.sender]
778     @post __post.accounts[_from] == accounts[_from]
779     @post __post.accounts[_to] == accounts[_to]
780     @post __post.accounts[feeCollector] == accounts[feeCollector]
781     @post __post.allowances[_from][msg.sender] == allowances[_from][msg.sender]
782     @post !__return
783 */
784 /*@CTK "transferFrom minVariableFee succeed"
785     @tag spec
786     @tag assume_completion
787     @pre !frozen
788     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
789         BLACK_LIST_FLAG)
790     @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
791         ZERO_FEE_FLAG)
792     @pre maxVariableFee > minVariableFee
793     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee
794     @pre (_value + fixedFee + minVariableFee) <= accounts[_from] && (_value + fixedFee
795         + minVariableFee) <= allowances [_from][msg.sender]
796     @post __post.accounts[_from] == accounts[_from] - (_value + fixedFee +
797         minVariableFee)
798     @post __post.accounts[_to] == accounts[_to] + _value
799     @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee +
800         minVariableFee)
801     @post __post.allowances[_from][msg.sender] == allowances[_from][msg.sender] - (
802         _value + fixedFee + minVariableFee)
803     @post __return
804 */
805 /*@CTK "transferFrom minVariableFee fail"
806     @tag spec
807     @tag assume_completion
808     @pre !frozen
809     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
810         BLACK_LIST_FLAG)
811     @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
812         ZERO_FEE_FLAG)
813     @pre maxVariableFee > minVariableFee
814     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee

```

```

801     @pre (_value + fixedFee + minVariableFee) > accounts[_from] || (_value + fixedFee
      + minVariableFee) > allowances [_from][msg.sender]
802     @post __post.accounts[_from] == accounts[_from]
803     @post __post.accounts[_to] == accounts[_to]
804     @post __post.accounts[feeCollector] == accounts[feeCollector]
805     @post __post.allowances[_from][msg.sender] == allowances[_from][msg.sender]
806     @post !__return
807 */
808 /*@CTK "transferFrom normal variableFee succeed"
809     @tag spec
810     @tag assume_completion
811     @pre !frozen
812     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
      BLACK_LIST_FLAG)
813     @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
      ZERO_FEE_FLAG)
814     @pre maxVariableFee > minVariableFee
815     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee
816     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee
817     @pre (_value + fixedFee + (_value * variableFeeNumerator) / FEE_DENOMINATOR) <=
      accounts[_from] && (_value + fixedFee + (_value * variableFeeNumerator) /
      FEE_DENOMINATOR) <= allowances [_from][msg.sender]
818     @post __post.accounts[_from] == accounts[_from] - (_value + fixedFee + (_value *
      variableFeeNumerator) / FEE_DENOMINATOR)
819     @post __post.accounts[_to] == accounts[_to] + _value
820     @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee + (
      _value * variableFeeNumerator) / FEE_DENOMINATOR)
821     @post __post.allowances[_from][msg.sender] == allowances[_from][msg.sender] - (
      _value + fixedFee + (_value * variableFeeNumerator) / FEE_DENOMINATOR)
822     @post __return
823 */
824 /*@CTK "transferFrom normal variableFee fail"
825     @tag spec
826     @tag assume_completion
827     @pre !frozen
828     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
      BLACK_LIST_FLAG)
829     @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
      ZERO_FEE_FLAG)
830     @pre maxVariableFee > minVariableFee
831     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee
832     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee
833     @pre (_value + fixedFee + (_value * variableFeeNumerator) / FEE_DENOMINATOR) >
      accounts[_from] || (_value + fixedFee + (_value * variableFeeNumerator) /
      FEE_DENOMINATOR) > allowances [_from][msg.sender]
834     @post __post.accounts[_from] == accounts[_from]
835     @post __post.accounts[_to] == accounts[_to]
836     @post __post.accounts[feeCollector] == accounts[feeCollector]
837     @post __post.allowances[_from][msg.sender] == allowances[_from][msg.sender]
838     @post !__return
839 */
840 function transferFrom (address _from, address _to, uint256 _value)
841 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
842     if (frozen) return false;
843     else if (
844         (addressFlags [_from] | addressFlags [_to]) & BLACK_LIST_FLAG ==
845         BLACK_LIST_FLAG)
846         return false;

```

```

847     else {
848         uint256 fee =
849             (addressFlags [_from] | addressFlags [_to]) & ZERO_FEE_FLAG == ZERO_FEE_FLAG ?
850             0 :
851             calculateFee (_value);
852
853         if (_value <= allowances [_from][msg.sender] &&
854             fee <= safeSub (allowances [_from][msg.sender], _value) &&
855             _value <= accounts [_from] &&
856             fee <= safeSub (accounts [_from], _value)) {
857             require (AbstractToken.transferFrom (_from, _to, _value));
858             require (AbstractToken.transferFrom (_from, feeCollector, fee));
859             return true;
860         } else return false;
861     }
862 }
863
864 /**
865  * Allow given spender to transfer given number of tokens from message sender.
866  *
867  * @param _spender address to allow the owner of to transfer tokens from
868  *         message sender
869  * @param _value number of tokens to allow to transfer
870  * @return true if token transfer was successfully approved, false otherwise
871  */
872 //@CTK NO_OVERFLOW
873 //@CTK NO_BUF_OVERFLOW
874 //@CTK NO_ASF
875 /*CTK "EURSToken approve"
876     @tag spec
877     @tag assume_completion
878     @post (__post.allowances[msg.sender][_spender]) == (_value)
879 */
880 function approve (address _spender, uint256 _value)
881 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool success) {
882     return AbstractToken.approve (_spender, _value);
883 }
884
885 /**
886  * Tell how many tokens given spender is currently allowed to transfer from
887  * given owner.
888  *
889  * @param _owner address to get number of tokens allowed to be transferred
890  *         from the owner of
891  * @param _spender address to get number of tokens allowed to be transferred
892  *         by the owner of
893  * @return number of tokens given spender is currently allowed to transfer
894  *         from given owner
895  */
896 //@CTK NO_OVERFLOW
897 //@CTK NO_BUF_OVERFLOW
898 //@CTK NO_ASF
899 /*CTK "EURSToken allowance"
900     @tag spec
901     @tag assume_completion
902     @post (remaining) == (allowances[_owner][_spender])
903 */
904 function allowance (address _owner, address _spender)

```

```

905 public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 remaining) {
906     return AbstractToken.allowance (_owner, _spender);
907 }
908
909 /**
910  * Transfer given number of token from the signed defined by digital signature
911  * to given recipient.
912  *
913  * @param _to address to transfer token to the owner of
914  * @param _value number of tokens to transfer
915  * @param _fee number of tokens to give to message sender
916  * @param _nonce nonce of the transfer
917  * @param _v parameter V of digital signature
918  * @param _r parameter R of digital signature
919  * @param _s parameter S of digital signature
920  */
921 /*@CTK "delegatedTransfer frozen"
922   @tag spec
923   @tag assume_completion
924   @pre frozen
925   @post (__post.accounts[_from] == accounts[_from])
926   @post (__post.accounts[_to] == accounts[_to])
927   @post (__post.accounts[feeCollector] == accounts[feeCollector])
928   @post (__post.accounts[msg.sender] == accounts[msg.sender])
929   @post __post._nonce[_from] == nonces[_from]
930   @post !__return
931 */
932 /*@CTK "delegatedTransfer signature reused"
933   @tag spec
934   @tag assume_completion
935   @pre !frozen
936   @pre _nonce != nonces [_from]
937   @pre (addressFlags[_from] == BLACK_LIST_FLAG) || (addressFlags[_to] ==
       BLACK_LIST_FLAG)
938   @post (__post.accounts[_from] == accounts[_from])
939   @post (__post.accounts[_to] == accounts[_to])
940   @post (__post.accounts[feeCollector] == accounts[feeCollector])
941   @post (__post.accounts[msg.sender] == accounts[msg.sender])
942   @post __post._nonce[_from] == nonces[_from]
943   @post !__return
944 */
945 /*@CTK "delegatedTransfer blacklisted"
946   @tag spec
947   @tag assume_completion
948   @pre !frozen
949   @pre _nonce == nonces [_from]
950   @pre (addressFlags[_from] == BLACK_LIST_FLAG) || (addressFlags[_to] ==
       BLACK_LIST_FLAG)
951   @post (__post.accounts[_from] == accounts[_from])
952   @post (__post.accounts[_to] == accounts[_to])
953   @post (__post.accounts[feeCollector] == accounts[feeCollector])
954   @post (__post.accounts[msg.sender] == accounts[msg.sender])
955   @post __post._nonce[_from] == nonces[_from]
956   @post !__return
957 */
958 /*@CTK "delegatedTransfer zero-feed succeed"
959   @tag spec
960   @tag assume_completion

```



```

961     @pre !frozen
962     @pre _nonce == nonces [_from]
963     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
          BLACK_LIST_FLAG)
964     @pre (addressFlags[_from] == ZERO_FEE_FLAG) || (addressFlags [_to] ==
          ZERO_FEE_FLAG)
965     @pre (_value + _fee <= accounts[_from])
966     @post (__post.accounts[_from] == accounts[_from] - _value - _fee)
967     @post (__post.accounts[_to] == accounts[_to] + _value + _fee)
968     @post (__post.accounts[feeCollector] == accounts[feeCollector])
969     @post (__post.accounts[msg.sender] == accounts[msg.sender] + _fee)
970     @post __post._nonce[_from] == nonces[_from] + 1
971     @post __return
972 */
973 /*@CTK "delegatedTransfer zero-feed fail"
974     @tag spec
975     @tag assume_completion
976     @pre !frozen
977     @pre _nonce == nonces [_from]
978     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
          BLACK_LIST_FLAG)
979     @pre (addressFlags[_from] == ZERO_FEE_FLAG) || (addressFlags [_to] ==
          ZERO_FEE_FLAG)
980     @pre (_value + _fee > accounts[_from])
981     @post (__post.accounts[_from] == accounts[_from])
982     @post (__post.accounts[_to] == accounts[_to])
983     @post (__post.accounts[feeCollector] == accounts[feeCollector])
984     @post (__post.accounts[msg.sender] == accounts[msg.sender])
985     @post __post._nonce[_from] == nonces[_from]
986     @post __return
987 */
988 /*@CTK "delegatedTransfer maxVariableFee succeed"
989     @tag spec
990     @tag assume_completion
991     @pre !frozen
992     @pre _nonce == nonces [_from]
993     @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
          BLACK_LIST_FLAG)
994     @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
          ZERO_FEE_FLAG)
995     @pre maxVariableFee > minVariableFee
996     @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee
997     @pre (_value + fixedFee + maxVariableFee + _fee) <= accounts[_from]
998     @post __post.accounts[_from] == accounts[_from] - (_value + fixedFee +
          maxVariableFee + _fee)
999     @post __post.accounts[_to] == accounts[_to] + _value
1000    @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee +
          maxVariableFee)
1001    @post (__post.accounts[msg.sender] == accounts[msg.sender] + _fee)
1002    @post __post._nonce[_from] == nonces[_from] + 1
1003    @post __return
1004 */
1005 /*@CTK "delegatedTransfer maxVariableFee fail"
1006     @tag spec
1007     @tag assume_completion
1008     @pre !frozen
1009     @pre _nonce == nonces [_from]

```



```

1010  @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
      BLACK_LIST_FLAG)
1011  @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
      ZERO_FEE_FLAG)
1012  @pre maxVariableFee > minVariableFee
1013  @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee
1014  @pre (_value + fixedFee + maxVariableFee + _fee) > accounts[_from]
1015  @post __post.accounts[_from] == accounts[_from]
1016  @post __post.accounts[_to] == accounts[_to]
1017  @post __post.accounts[feeCollector] == accounts[feeCollector]
1018  @post (__post.accounts[msg.sender] == accounts[msg.sender])
1019  @post __post._nonce[_from] == nonces[_from]
1020  @post !__return
1021  */
1022  /*@CTK "delegatedTransfer minVariableFee succeed"
1023  @tag spec
1024  @tag assume_completion
1025  @pre !frozen
1026  @pre _nonce == nonces [_from]
1027  @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
      BLACK_LIST_FLAG)
1028  @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
      ZERO_FEE_FLAG)
1029  @pre maxVariableFee > minVariableFee
1030  @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee
1031  @pre (_value + fixedFee + minVariableFee + _fee) <= accounts[_from]
1032  @post __post.accounts[_from] == accounts[_from] - (_value + fixedFee +
      minVariableFee + _fee)
1033  @post __post.accounts[_to] == accounts[_to] + _value
1034  @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee +
      minVariableFee)
1035  @post (__post.accounts[msg.sender] == accounts[msg.sender] + _fee)
1036  @post __post._nonce[_from] == nonces[_from] + 1
1037  @post __return
1038  */
1039  /*@CTK "delegatedTransfer minVariableFee fail"
1040  @tag spec
1041  @tag assume_completion
1042  @pre !frozen
1043  @pre _nonce == nonces [_from]
1044  @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
      BLACK_LIST_FLAG)
1045  @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
      ZERO_FEE_FLAG)
1046  @pre maxVariableFee > minVariableFee
1047  @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee
1048  @pre (_value + fixedFee + minVariableFee + _fee) > accounts[_from]
1049  @post __post.accounts[_from] == accounts[_from]
1050  @post __post.accounts[_to] == accounts[_to]
1051  @post __post.accounts[feeCollector] == accounts[feeCollector]
1052  @post (__post.accounts[msg.sender] == accounts[msg.sender])
1053  @post __post._nonce[_from] == nonces[_from]
1054  @post !__return
1055  */
1056  /*@CTK "delegatedTransfer normal variableFee succeed"
1057  @tag spec
1058  @tag assume_completion
1059  @pre !frozen

```

```

1060 @pre _nonce == nonces [_from]
1061 @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
    BLACK_LIST_FLAG)
1062 @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
    ZERO_FEE_FLAG)
1063 @pre maxVariableFee > minVariableFee
1064 @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee
1065 @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee
1066 @pre (_value + fixedFee + (_value * variableFeeNumerator) / FEE_DENOMINATOR + _fee
    ) <= accounts[_from]
1067 @post __post.accounts[_from] == accounts[_from] - (_value + fixedFee + (_value *
    variableFeeNumerator) / FEE_DENOMINATOR + _fee)
1068 @post __post.accounts[_to] == accounts[_to] + _value
1069 @post __post.accounts[feeCollector] == accounts[feeCollector] + (fixedFee + (
    _value * variableFeeNumerator) / FEE_DENOMINATOR)
1070 @post (__post.accounts[msg.sender] == accounts[msg.sender] + _fee)
1071 @post __post._nonce[_from] == nonces[_from] + 1
1072 @post __return
1073 */
1074 /*@CTK "delegatedTransfer normal variableFee fail"
1075 @tag spec
1076 @tag assume_completion
1077 @pre !frozen
1078 @pre _nonce == nonces [_from]
1079 @pre (addressFlags[_from] != BLACK_LIST_FLAG) && (addressFlags [_to] !=
    BLACK_LIST_FLAG)
1080 @pre (addressFlags[_from] != ZERO_FEE_FLAG) && (addressFlags [_to] !=
    ZERO_FEE_FLAG)
1081 @pre maxVariableFee > minVariableFee
1082 @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee
1083 @pre (_value * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee
1084 @pre (_value + fixedFee + (_value * variableFeeNumerator) / FEE_DENOMINATOR + _fee
    ) > accounts[_from]
1085 @post __post.accounts[_from] == accounts[_from]
1086 @post __post.accounts[_to] == accounts[_to]
1087 @post __post.accounts[feeCollector] == accounts[feeCollector]
1088 @post (__post.accounts[msg.sender] == accounts[msg.sender])
1089 @post __post._nonce[_from] == nonces[_from]
1090 @post !__return
1091 */
1092 function delegatedTransfer (
1093     address _to, uint256 _value, uint256 _fee,
1094     uint256 _nonce, uint8 _v, bytes32 _r, bytes32 _s)
1095 public delegatable payable returns (bool) {
1096     if (frozen) return false;
1097     else {
1098         address _from = ecrecover (
1099             keccak256 (
1100                 thisAddress (), messageSenderAddress (), _to, _value, _fee, _nonce),
1101             _v, _r, _s);
1102
1103         if (_nonce != nonces [_from]) return false;
1104
1105         if (
1106             (addressFlags [_from] | addressFlags [_to]) & BLACK_LIST_FLAG ==
1107             BLACK_LIST_FLAG)
1108             return false;
1109     }

```

```

1110     uint256 fee =
1111         (addressFlags [_from] | addressFlags [_to]) & ZERO_FEE_FLAG == ZERO_FEE_FLAG ?
1112         0 :
1113         calculateFee (_value);
1114
1115     uint256 balance = accounts [_from];
1116     if (_value > balance) return false;
1117     balance = safeSub (balance, _value);
1118     if (fee > balance) return false;
1119     balance = safeSub (balance, fee);
1120     if (_fee > balance) return false;
1121     balance = safeSub (balance, _fee);
1122
1123     nonces [_from] = _nonce + 1;
1124
1125     accounts [_from] = balance;
1126     accounts [_to] = safeAdd (accounts [_to], _value);
1127     accounts [feeCollector] = safeAdd (accounts [feeCollector], fee);
1128     accounts [msg.sender] = safeAdd (accounts [msg.sender], _fee);
1129
1130     Transfer (_from, _to, _value);
1131     Transfer (_from, feeCollector, fee);
1132     Transfer (_from, msg.sender, _fee);
1133
1134     return true;
1135 }
1136 }
1137
1138 /**
1139  * Create tokens.
1140  *
1141  * @param _value number of tokens to be created.
1142  */
1143 // @CTK FAIL NO_OVERFLOW
1144 // @CTK NO_BUF_OVERFLOW
1145 // @CTK FAIL NO_ASF
1146 /* @CTK "EURSToken createTokens"
1147    @tag assume_completion
1148    @pre msg.sender == owner
1149    @post ((_value > 0) && (_value > MAX_TOKENS_COUNT - tokensCount)) -> !__return
1150    @post ((_value == 0) || (_value <= MAX_TOKENS_COUNT - tokensCount)) -> __return
1151    @post ((_value == 0) && (_value <= MAX_TOKENS_COUNT - tokensCount)) -> (__post.
1152        accounts[msg.sender] == accounts[msg.sender] + _value) && (__post.tokensCount
1153        == tokensCount + _value)
1154 */
1155 function createTokens (uint256 _value)
1156 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1157     require (msg.sender == owner);
1158
1159     if (_value > 0) {
1160         if (_value <= safeSub (MAX_TOKENS_COUNT, tokensCount)) {
1161             accounts [msg.sender] = safeAdd (accounts [msg.sender], _value);
1162             tokensCount = safeAdd (tokensCount, _value);
1163
1164             Transfer (address (0), msg.sender, _value);
1165
1166             return true;
1167         } else return false;
1168     }

```

```

1166     } else return true;
1167 }
1168
1169 /**
1170  * Burn tokens.
1171  *
1172  * @param _value number of tokens to burn
1173  */
1174 //@CTK NO_OVERFLOW
1175 //@CTK NO_BUF_OVERFLOW
1176 //@CTK FAIL_NO_ASF
1177 /*@CTK "EURSToken burnTokens"
1178   @tag assume_completion
1179   @pre msg.sender == owner
1180   @post ((_value > 0) && (_value > accounts [msg.sender])) -> !__return
1181   @post ((_value == 0) || (_value <= accounts [msg.sender])) -> __return
1182   @post ((_value == 0) || (_value <= accounts [msg.sender])) -> (__post.accounts[msg
        .sender] == accounts[msg.sender] - _value) && (__post.tokensCount ==
        tokensCount - _value)
1183 */
1184 function burnTokens (uint256 _value)
1185 public /*>IGNORE delegatable IGNORE<*/ payable returns (bool) {
1186     require (msg.sender == owner);
1187
1188     if (_value > 0) {
1189         if (_value <= accounts [msg.sender]) {
1190             accounts [msg.sender] = safeSub (accounts [msg.sender], _value);
1191             tokensCount = safeSub (tokensCount, _value);
1192
1193             Transfer (msg.sender, address (0), _value);
1194
1195             return true;
1196         } else return false;
1197     } else return true;
1198 }
1199
1200 /**
1201  * Freeze token transfers.
1202  */
1203 //@CTK NO_OVERFLOW
1204 //@CTK NO_BUF_OVERFLOW
1205 //@CTK NO_ASF
1206 /*@CTK "freezeTransfers"
1207   @tag assume_completion
1208   @pre msg.sender == owner
1209   @post __post.frozen
1210 */
1211 function freezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1212     require (msg.sender == owner);
1213
1214     if (!frozen) {
1215         frozen = true;
1216
1217         Freeze ();
1218     }
1219 }
1220
1221 /**

```

```
1222 * Unfreeze token transfers.
1223 */
1224 //@CTK NO_OVERFLOW
1225 //@CTK NO_BUF_OVERFLOW
1226 //@CTK NO_ASF
1227 /*CTK "unfreezeTransfers"
1228   @tag assume_completion
1229   @pre msg.sender == owner
1230   @post __post.frozen == false
1231 */
1232 function unfreezeTransfers () public /*>IGNORE delegatable IGNORE<*/ payable {
1233   require (msg.sender == owner);
1234
1235   if (frozen) {
1236     frozen = false;
1237
1238     Unfreeze ();
1239   }
1240 }
1241
1242 /**
1243  * Set smart contract owner.
1244  *
1245  * @param _newOwner address of the new owner
1246  */
1247 //@CTK NO_OVERFLOW
1248 //@CTK NO_BUF_OVERFLOW
1249 //@CTK NO_ASF
1250 /*CTK "unfreezeTransfers"
1251   @tag assume_completion
1252   @post msg.sender == owner
1253   @post __post.owner == _newOwner
1254 */
1255 function setOwner (address _newOwner) public {
1256   require (msg.sender == owner);
1257
1258   owner = _newOwner;
1259 }
1260
1261 /**
1262  * Set fee collector.
1263  *
1264  * @param _newFeeCollector address of the new fee collector
1265  */
1266 //@CTK NO_OVERFLOW
1267 //@CTK NO_BUF_OVERFLOW
1268 //@CTK NO_ASF
1269 /*CTK "setFeeCollector"
1270   @tag assume_completion
1271   @pre msg.sender == owner
1272   @post (__post.feeCollector) == (_newFeeCollector)
1273 */
1274 function setFeeCollector (address _newFeeCollector)
1275 public /*>IGNORE delegatable IGNORE<*/ payable {
1276   require (msg.sender == owner);
1277
1278   feeCollector = _newFeeCollector;
1279 }
```

```

1280
1281 /**
1282  * Get current nonce for token holder with given address, i.e. nonce this
1283  * token holder should use for next delegated transfer.
1284  *
1285  * @param _owner address of the token holder to get nonce for
1286  * @return current nonce for token holder with give address
1287  */
1288 //@CTK NO_OVERFLOW
1289 //@CTK NO_BUF_OVERFLOW
1290 //@CTK NO_ASF
1291 /*CTK nonce
1292  @tag assume_completion
1293  @post __return == nonces [_owner]
1294 */
1295 function nonce (address _owner) public view /*>IGNORE delegatable IGNORE<*/ returns
1296     (uint256) {
1297     return nonces [_owner];
1298 }
1299
1300 /**
1301  *
1302  * @param _fixedFee fixed fee in token units
1303  * @param _minVariableFee minimum variable fee in token units
1304  * @param _maxVariableFee maximum variable fee in token units
1305  * @param _variableFeeNumerator variable fee numerator
1306  */
1307 //@CTK NO_OVERFLOW
1308 //@CTK NO_BUF_OVERFLOW
1309 //@CTK NO_ASF
1310 /*CTK setFeeCollector
1311  @tag assume_completion
1312  @pre msg.sender == owner
1313  @pre _minVariableFee <= _maxVariableFee
1314  @pre _variableFeeNumerator <= MAX_FEE_NUMERATOR
1315  @post (__post.fixedFee) == (_fixedFee)
1316  @post (__post.minVariableFee) == (_minVariableFee)
1317  @post (__post.maxVariableFee) == (_maxVariableFee)
1318  @post (__post.variableFeeNumerator) == (_variableFeeNumerator)
1319 */
1320 function setFeeParameters (
1321     uint256 _fixedFee,
1322     uint256 _minVariableFee,
1323     uint256 _maxVariableFee,
1324     uint256 _variableFeeNumerator) public /*>IGNORE delegatable IGNORE<*/ payable {
1325     require (msg.sender == owner);
1326
1327     require (_minVariableFee <= _maxVariableFee);
1328     require (_variableFeeNumerator <= MAX_FEE_NUMERATOR);
1329
1330     fixedFee = _fixedFee;
1331     minVariableFee = _minVariableFee;
1332     maxVariableFee = _maxVariableFee;
1333     variableFeeNumerator = _variableFeeNumerator;
1334
1335     FeeChange (
1336         _fixedFee, _minVariableFee, _maxVariableFee, _variableFeeNumerator);

```

```

1337 }
1338
1339 /**
1340  * Get fee parameters.
1341  *
1342  * @return fee parameters
1343  */
1344 //@CTK NO_OVERFLOW
1345 //@CTK NO_BUF_OVERFLOW
1346 //@CTK NO_ASF
1347 /*CTK getFeeParameters
1348  @tag assume_completion
1349  @post _fixedFee == fixedFee
1350  @post _minVariableFee == minVariableFee
1351  @post _maxVariableFee == maxVariableFee
1352  @post _variableFeeNumnerator == variableFeeNumerator
1353 */
1354 function getFeeParameters () public /*>IGNORE delegatable IGNORE<*/ view returns (
1355     uint256 _fixedFee,
1356     uint256 _minVariableFee,
1357     uint256 _maxVariableFee,
1358     uint256 _variableFeeNumnerator) {
1359     _fixedFee = fixedFee;
1360     _minVariableFee = minVariableFee;
1361     _maxVariableFee = maxVariableFee;
1362     _variableFeeNumnerator = variableFeeNumerator;
1363 }
1364
1365 /**
1366  * Calculate fee for transfer of given number of tokens.
1367  *
1368  * @param _amount transfer amount to calculate fee for
1369  * @return fee for transfer of given amount
1370  */
1371 //@CTK FAIL NO_OVERFLOW
1372 //@CTK NO_BUF_OVERFLOW
1373 //@CTK FAIL NO_ASF
1374 /*CTK calculateFee
1375  @tag assume_completion
1376  @pre _amount <= MAX_TOKENS_COUNT
1377  @pre maxVariableFee > minVariableFee
1378  @post (_amount * variableFeeNumerator) / FEE_DENOMINATOR > maxVariableFee -> _fee
1379      == fixedFee + maxVariableFee
1380  @post (_amount * variableFeeNumerator) / FEE_DENOMINATOR < minVariableFee -> _fee
1381      == fixedFee + minVariableFee
1382  @post ((_amount * variableFeeNumerator) / FEE_DENOMINATOR >= minVariableFee && (
1383      _amount * variableFeeNumerator) / FEE_DENOMINATOR <= maxVariableFee) -> _fee
1384      == fixedFee + (_amount * variableFeeNumerator) / FEE_DENOMINATOR
1385 */
1386 function calculateFee (uint256 _amount)
1387     public /*>IGNORE delegatable IGNORE<*/ view returns (uint256 _fee) {
1388     require (_amount <= MAX_TOKENS_COUNT);
1389
1390     _fee = safeMul (_amount, variableFeeNumerator) / FEE_DENOMINATOR;
1391     if (_fee < minVariableFee) _fee = minVariableFee;
1392     if (_fee > maxVariableFee) _fee = maxVariableFee;
1393     _fee = safeAdd (_fee, fixedFee);
1394 }

```

```

1391
1392 /**
1393  * Set flags for given address.
1394  *
1395  * @param _address address to set flags for
1396  * @param _flags flags to set
1397  */
1398 //@CTK NO_OVERFLOW
1399 //@CTK NO_BUF_OVERFLOW
1400 //@CTK NO_ASF
1401 /*CTK setFlags
1402  @tag assume_completion
1403  @pre msg.sender == owner
1404  @post __post.addressFlags [_address] == _flags
1405  */
1406 function setFlags (address _address, uint256 _flags)
1407 public /*>IGNORE delegatable IGNORE<*/ payable {
1408     require (msg.sender == owner);
1409
1410     addressFlags [_address] = _flags;
1411 }
1412
1413 /**
1414  * Get flags for given address.
1415  *
1416  * @param _address address to get flags for
1417  * @return flags for given address
1418  */
1419 function flags (address _address) public delegatable view returns (uint256) {
1420     return addressFlags [_address];
1421 }
1422
1423 /**
1424  * Set address of smart contract to delegate execution of delegatable methods
1425  * to.
1426  *
1427  * @param _delegate address of smart contract to delegate execution of
1428  * delegatable methods to, or zero to not delegate delegatable methods
1429  * execution.
1430  */
1431 //@CTK NO_OVERFLOW
1432 //@CTK NO_BUF_OVERFLOW
1433 //@CTK NO_ASF
1434 /*CTK "setDelegate"
1435  @tag assume_completion
1436  @pre msg.sender == owner
1437  @post __post.delegate == _delegate
1438  */
1439 function setDelegate (address _delegate) public {
1440     require (msg.sender == owner);
1441
1442     if (delegate != _delegate) {
1443         delegate = _delegate;
1444         Delegation (delegate);
1445     }
1446 }
1447
1448 /**

```



```
1449  * Get address of this smart contract.
1450  *
1451  * @return address of this smart contract
1452  */
1453  function thisAddress () internal view returns (address) {
1454      return this;
1455  }
1456
1457  /**
1458   * Get address of message sender.
1459   *
1460   * @return address of this smart contract
1461   */
1462  function messageSenderAddress () internal view returns (address) {
1463      return msg.sender;
1464  }
1465
1466  /**
1467   * Owner of the smart contract.
1468   */
1469  address internal owner;
1470
1471  /**
1472   * Address where fees are sent to.
1473   */
1474  address internal feeCollector;
1475
1476  /**
1477   * Number of tokens in circulation.
1478   */
1479  uint256 internal tokensCount;
1480
1481  /**
1482   * Whether token transfers are currently frozen.
1483   */
1484  bool internal frozen;
1485
1486  /**
1487   * Mapping from sender's address to the next delegated transfer nonce.
1488   */
1489  mapping (address => uint256) internal nonces;
1490
1491  /**
1492   * Fixed fee amount in token units.
1493   */
1494  uint256 internal fixedFee;
1495
1496  /**
1497   * Minimum variable fee in token units.
1498   */
1499  uint256 internal minVariableFee;
1500
1501  /**
1502   * Maximum variable fee in token units.
1503   */
1504  uint256 internal maxVariableFee;
1505
1506  /**
```

```

1507  * Variable fee numerator.
1508  */
1509  uint256 internal variableFeeNumerator;
1510
1511  /**
1512   * Maps address to its flags.
1513   */
1514  mapping (address => uint256) internal addressFlags;
1515
1516  /**
1517   * Address of smart contract to delegate execution of delegatable methods to,
1518   * or zero to not delegate delegatable methods execution.
1519   */
1520  address internal delegate;
1521
1522  /**
1523   * Logged when token transfers were frozen.
1524   */
1525  event Freeze ();
1526
1527  /**
1528   * Logged when token transfers were unfrozen.
1529   */
1530  event Unfreeze ();
1531
1532  /**
1533   * Logged when fee parameters were changed.
1534   *
1535   * @param fixedFee fixed fee in token units
1536   * @param minVariableFee minimum variable fee in token units
1537   * @param maxVariableFee maximum variable fee in token units
1538   * @param variableFeeNumerator variable fee numerator
1539   */
1540  event FeeChange (
1541      uint256 fixedFee,
1542      uint256 minVariableFee,
1543      uint256 maxVariableFee,
1544      uint256 variableFeeNumerator);
1545
1546  /**
1547   * Logged when address of smart contract execution of delegatable methods is
1548   * delegated to was changed.
1549   *
1550   * @param delegate new address of smart contract execution of delegatable
1551   * methods is delegated to or zero if execution of delegatable methods is
1552   * not delegated.
1553   */
1554  event Delegation (address delegate);
1555  }

```