

CERTIK AUDIT REPORT FOR USDK



Request Date: 2018-04-03

Revision Date: 2019-06-09

Platform Name:



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	12
How to read	12
Source Code with CertiK Labels	26

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 OKEx (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 USDK. This audit was conducted to discover issues and vulnerabilities in the source code of USDK'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 09, 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.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

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

Vulnerability Details

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Scope of Work

CertiK team is invited by OKEx to audit the design and implementations of its to be released stable coin USDK smart contract. The goal of this audit is to review USDK's solidity implementation on top of its business logic, detect potential security vulnerabilities, understand its general design and architecture, and uncover bugs that could compromise the system in production environment.

Aside from manually reviews by smart contract experts, the USDK smart contracts have been analyzed under different perspectives and with different tools including CertiK's proprietary formal verification platform as well as static analysis system to mathematically ensure that the logic works as intended. During the process, CertiK team has been actively interacting with USDK engineers when there was any potential loopholes or recommended design changes during the audit process, and USDK team has been actively giving updates for the source code and feedback about the business logics.

Overall we found the USDK and the proxy contracts follow good practices, the business logics and intentions are reasonable given its stable coin nature. With the final updated version, we did not find any potential security risks for the core token features and add-on functionalities, however token holders should still be aware of the administrative authority of the multiple admin roles (described in a following section), who are able to perform critical actions such as pause, freeze and increase/decrease token supply.

We urge more unit test cases to be added into the repository to simulate different scenarios especially the use case to upgrade to a new token via the unstructured proxy pattern. Meanwhile, it is recommended to have a more well-detailed document for the public to describe the source code specifications and implementations.

Token Overview

The USDK is a token smart contract with stable coin model implementations, with the purpose to anchor US Dollar in its reserve. Extra logics were introduced, for the nature of stable coin features, to the smart contract as well for the purpose of central governing, compliance and ability for future upgrades.

On top of the ERC20 standard implementations, we believe it's important to describe the roles together with their capacities and business intentions in the smart contract.

- **owner** – able to pause and unpaue for token balance transfers or allowance approvals; able to assign `supplyController` and `lawEnforcementRole` with new addresses. It possesses all powers from those two roles.
- **supplyController** – able to increase or decrease the token total supply, with the ability to transfer itself to a new address.
- **lawEnforcementRole** – able to freeze, unfreeze and wipe out an address; it means to pause all actions for a target address, and is able to set its balance to 0 and decrease the total supply.
- **Proxy Admin** – able to upgrade the token address to a new one, which means the abandon of the original token implementations.

Considering that USDK is a compliance based stablecoin, the employment of Know-Your-Coin tracing ability would enhance the uniqueness of its value in the current market. Thus we believe adding the logging functionality would further benefit and shed insights on parties such as compliance officers, auditors, analysts, etc. The value of event logging includes but not limited to knowing which addresses trigger the methods even when the addresses do not meet the required condition.

Source Code SHA-256 Checksum

- **SafeMath.sol**
49cba5e8c9434328265b435d41c995f3b82337178c66b831c4322690a8f22ddf
- **Proxy.sol**
1d11e87a563e72820f113f0646ceb6cafc5506b94a8b080776449123442d59bd
- **UpgradeabilityProxy.sol**
ee13c4748a1156123e51948eb8e79b000c3d9eda168b7b02cd19dd0b5597b21a
- **OwedUpgradeabilityProxy.sol**
034f5f5450d28bf9852cd1181e40fdd5598ecb8ff3cda43ddb8297c351d7e6b1
- **OKUSDimpl.sol**
bde6a8fde008e97116d4937d41c217921a2c6a30430f9406939d3e27e94dadbf

Recommendations

The items below are notes from the CertiK team in accordance to our audit. These suggestions are optional, and may have low impact to the overall aspects of the USDK smart contracts. As such, these are optional edits for USDK to consider for enhancement.

OKUSDimpl.sol

- **setSupplyController()** – Check the supplyController & any newSupplyController address is not a frozen address.
- **transferOwnership()** – Check the `_newOwner` address is not a frozen address.
- **setLawEnforcementRole()** – Check the `_newLawEnforcementRole` address is not a frozen address.
- **increaseSupply(), decreaseSupply()** – Please check and enforce that all the token should not allow to mint, burn or transfer to the supplyController address while pause state.
- **isDeprecated()** – Considering having a state variable `_isDeprecated`, when upgraded to a new token contract, set the value to `true` to prevent any further actions on the contract. Currently we can leverage `pausable` to achieve similar functionality.

OwnedUpgradeabilityProxy.sol, UpgradeabilityProxy.sol

- **keccak256()** – Consider to assign the constant proxyOwnerPosition to its hash representation, also note that the parameter for the keccak256 could be any random string, i.e. `okusd.proxy.owner` OR `okusd.proxy.implementation`.

Best practice

The checklist below helps to evaluate the general quality of the solidity project.

Solidity Protocol

- ✓ Use latest stable solidity version
- ✓ Handle possible errors properly when making external calls
- ✓ Provide error message along with require()
- ✓ Use modifiers properly
- ✓ Use events to monitor contract activities
- ✓ Refer and use libraries properly
- ✓ No compiler warnings

Privilege Control

- ✓ Provide stop functionality for control and emergency handling
- ✓ Restrict access to sensitive functions

Documentation

- Provide project documentation and execution guidance
- ✓ Provide inline comment for function intention
- ✓ Provide instruction to initialize and execute the test files

Testing

- Provide migration scripts
- Provide test scripts and coverage for potential scenarios

With the final update of the source code and delivery of the audit report, CertiK is able to conclude that the USDK contract is not vulnerable to any classically known anti-patterns or security issues.

While this CertiK review is a strong and positive indication, the audit report itself is not necessarily a guarantee of correctness or trustworthiness. CertiK always recommends seeking multiple opinions, test coverage, sandbox deployments before any mainnet release.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File USDKimpl.sol

```
1 pragma solidity ^0.4.24;
```

! Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
1 pragma solidity ^0.4.0;
```

! Version to compile has the following bug: 0.4.0: IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction, IdentityPrecompileReturnIgnored, HighOrderByteCleanStorage, OptimizerStaleKnowledgeAboutSHA3, LibrariesNotCallableFromPayableFunctions 0.4.1: IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction, IdentityPrecompileReturnIgnored, HighOrderByteCleanStorage, OptimizerStaleKnowledgeAboutSHA3, LibrariesNotCallableFromPayableFunctions 0.4.10: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction 0.4.11: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral 0.4.12: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput 0.4.13: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput 0.4.14: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue 0.4.15: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.16: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.17: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData,

NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.18: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.19: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.2: IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction, IdentityPrecompileReturnIgnored, HighOrderByteCleanStorage, OptimizerStaleKnowledgeAboutSHA3 0.4.20: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2 0.4.3: IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction, IdentityPrecompileReturnIgnored, HighOrderByteCleanStorage 0.4.4: IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction, IdentityPrecompileReturnIgnored 0.4.5: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction, IdentityPrecompileReturnIgnored, OptimizerStateKnowledgeNotResetForJumpdest 0.4.6: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction, IdentityPrecompileReturnIgnored 0.4.7: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction 0.4.8: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction 0.4.9: Unini-

tializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput, SkipEmptyStringLiteral, ConstantOptimizerSubtraction

INSECURE_COMPILER_VERSION

Line 1 in File Migrations.sol

```
1 pragma solidity ^0.4.17;
```



! Version to compile has the following bug: 0.4.17: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.18: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.19: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.20: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address


Verification date	 20, Oct 2018
Verification timespan	 395.38 ms

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

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

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

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

Counterexample	 This code violates the specification	
Initial environment	1	Counter Example:
	2	Before Execution:
	3	Input = {
	4	from = 0x0
	5	to = 0x0
	6	tokens = 0x6c
	7	}
	8	This = 0
	52	}
	53	balance: 0x0
	54	}
	55	}
Post environment	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

Formal Verification Request 1

initialize

📅 09, Jun 2019

🕒 32.36 ms

Line 113-121 in File USDKimpl.sol

```
113  /*@CTK initialize
114     @tag assume_completion
115     @post !initialized
116     @post __post.initialized
117     @post __post.owner == msg.sender
118     @post __post.lawEnforcementRole == address(0)
119     @post __post.totalSupply_ == 0
120     @post __post.supplyController == msg.sender
121  */
```

Line 122-129 in File USDKimpl.sol

```
122  function initialize() public {
123      require(!initialized, "already initialized");
124      owner = msg.sender;
125      lawEnforcementRole = address(0);
126      totalSupply_ = 0;
127      supplyController = msg.sender;
128      initialized = true;
129  }
```

✅ The code meets the specification

Formal Verification Request 2

OKUSDImplementation

📅 09, Jun 2019

🕒 114.03 ms

Line 137-144 in File USDKimpl.sol

```
137  /*@CTK OKUSDImplementation
138     @tag assume_completion
139     @post !initialized && !paused
140     @post __post.initialized && __post.paused
141     @post __post.lawEnforcementRole == address(0)
142     @post __post.totalSupply_ == 0
143     @post __post.supplyController == msg.sender
144  */
```

Line 145-148 in File USDKimpl.sol

```
145  constructor() public {
146      initialize();
147      pause();
148  }
```

✅ The code meets the specification

Formal Verification Request 3

totalSupply

📅 09, Jun 2019

🕒 5.35 ms

Line 155-157 in File USDKimpl.sol

```
155  /*@CTK totalSupply
156      @post __return == totalSupply_
157  */
```

Line 158-160 in File USDKimpl.sol

```
158  function totalSupply() public view returns (uint256) {
159      return totalSupply_;
160  }
```

✅ The code meets the specification

Formal Verification Request 4

transfer

📅 09, Jun 2019

🕒 270.42 ms

Line 167-176 in File USDKimpl.sol

```
167  /*@CTK transfer
168      @tag assume_completion
169      @pre msg.sender != _to
170      @post !paused
171      @post _to != address(0)
172      @post !frozen[_to] && !frozen[msg.sender]
173      @post _value <= balances[msg.sender]
174      @post __post.balances[msg.sender] == balances[msg.sender] - _value
175      @post __post.balances[_to] == balances[_to] + _value
176  */
```

Line 177-186 in File USDKimpl.sol

```
177  function transfer(address _to, uint256 _value) public whenNotPaused returns (bool)
178  {
179      require(_to != address(0), "cannot transfer to address zero");
180      require(!frozen[_to] && !frozen[msg.sender], "address frozen");
181      require(_value <= balances[msg.sender], "insufficient funds");
182
183      balances[msg.sender] = balances[msg.sender].sub(_value);
184      balances[_to] = balances[_to].add(_value);
185      emit Transfer(msg.sender, _to, _value);
186      return true;
187  }
```

✅ The code meets the specification

Formal Verification Request 5

balanceOf

📅 09, Jun 2019

🕒 4.49 ms

Line 193-195 in File USDKimpl.sol

```

193  /*@CTK balanceOf
194      @post __return == __post.balances[_addr]
195  */

```

Line 196-198 in File USDKimpl.sol

```

196  function balanceOf(address _addr) public view returns (uint256) {
197      return balances[_addr];
198  }

```

✅ The code meets the specification

Formal Verification Request 6

transferFrom

📅 09, Jun 2019

🕒 362.14 ms

Line 208-218 in File USDKimpl.sol

```

208  /*@CTK transferFrom
209      @tag assume_completion
210      @pre _to != _from && _to != msg.sender && _from != msg.sender
211      @post !paused
212      @post !frozen[_to] && !frozen[msg.sender] && !frozen[_from]
213      @post _value <= balances[_from]
214      @post _value <= _allowed[_from][msg.sender]
215      @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
216      @post __post.balances[_to] == balances[_to] + _value
217      @post __post.balances[_from] == balances[_from] - _value
218  */

```

Line 219-231 in File USDKimpl.sol

```

219  function transferFrom(address _from, address _to, uint256 _value) public
220      whenNotPaused returns (bool)
221  {
222      require(_to != address(0), "cannot transfer to address zero");
223      require(!frozen[_to] && !frozen[_from] && !frozen[msg.sender], "address frozen");
224      require(_value <= balances[_from], "insufficient funds");
225      require(_value <= _allowed[_from][msg.sender], "insufficient allowance");
226      balances[_from] = balances[_from].sub(_value);
227      balances[_to] = balances[_to].add(_value);
228      _allowed[_from][msg.sender] = _allowed[_from][msg.sender].sub(_value);
229      emit Transfer(_from, _to, _value);
230      return true;
231  }

```

✓ The code meets the specification

Formal Verification Request 7

approve

📅 09, Jun 2019

🕒 28.05 ms

Line 242-247 in File USDKimpl.sol

```
242  /*@CTK approve
243     @tag assume_completion
244     @post !paused
245     @post !frozen[_spender] && !frozen[msg.sender]
246     @post __post._allowed[msg.sender][_spender] == _value
247  */
```

Line 248-253 in File USDKimpl.sol

```
248  function approve(address _spender, uint256 _value) public whenNotPaused returns (
249      bool) {
250      require(!frozen[_spender] && !frozen[msg.sender], "address frozen");
251      _allowed[msg.sender][_spender] = _value;
252      emit Approval(msg.sender, _spender, _value);
253      return true;
254  }
```

✓ The code meets the specification

Formal Verification Request 8

_approve

📅 09, Jun 2019

🕒 28.07 ms

Line 261-266 in File USDKimpl.sol

```
261  /*@CTK _approve
262     @tag assume_completion
263     @post !frozen[spender] && !frozen[_owner]
264     @post spender != address(0) && _owner != address(0)
265     @post __post._allowed[_owner][spender] == value
266  */
```

Line 267-272 in File USDKimpl.sol

```
267  function _approve(address _owner, address spender, uint256 value) internal {
268      require(!frozen[spender] && !frozen[_owner], "address frozen");
269      require(spender != address(0) && _owner != address(0), "not address(0)");
270      _allowed[_owner][spender] = value;
271      emit Approval(_owner, spender, value);
272  }
```

✓ The code meets the specification

Formal Verification Request 9

allowance

📅 09, Jun 2019

🕒 7.01 ms

Line 280-282 in File USDKimpl.sol

```
280 /*@CTK allowance
281     @post __return == _allowed[_owner][_spender]
282 */
```

Line 283-286 in File USDKimpl.sol

```
283 function allowance(address _owner, address _spender) public view returns (uint256)
284 {
285     return _allowed[_owner][_spender];
286 }
```

✅ The code meets the specification

Formal Verification Request 10

increaseAllowance

📅 09, Jun 2019

🕒 71.44 ms

Line 298-305 in File USDKimpl.sol

```
298 /*@CTK increaseAllowance
299     @tag assume_completion
300     @post !paused
301     @post !frozen[spender] && !frozen[msg.sender]
302     @post spender != address(0) && msg.sender != address(0)
303     @post __post._allowed[msg.sender][spender] ==
304         _allowed[msg.sender][spender] + addedValue
305 */
```

Line 306-309 in File USDKimpl.sol

```
306 function increaseAllowance(address spender, uint256 addedValue) public
307     whenNotPaused returns (bool) {
308     _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
309     return true;
310 }
```

✅ The code meets the specification

Formal Verification Request 11

decreaseAllowance

📅 09, Jun 2019

🕒 64.54 ms

Line 321-328 in File USDKimpl.sol

```

321  /*@CTK decreaseAllowance
322     @tag assume_completion
323     @post !paused
324     @post !frozen[spender] && !frozen[msg.sender]
325     @post spender != address(0) && msg.sender != address(0)
326     @post __post._allowed[msg.sender][spender] ==
327         _allowed[msg.sender][spender] - subtractedValue
328  */

```

Line 329-332 in File USDKimpl.sol

```

329  function decreaseAllowance(address spender, uint256 subtractedValue) public
      whenNotPaused returns (bool) {
330      _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
          ));
331      return true;
332  }

```

✓ The code meets the specification

Formal Verification Request 12

transferOwnership

📅 09, Jun 2019

🕒 25.0 ms

Line 348-353 in File USDKimpl.sol

```

348  /*@CTK transferOwnership
349     @tag assume_completion
350     @post owner == msg.sender
351     @post _newOwner != address(0)
352     @post __post.owner == _newOwner
353  */

```

Line 354-358 in File USDKimpl.sol

```

354  function transferOwnership(address _newOwner) public onlyOwner {
355      require(_newOwner != address(0), "cannot transfer ownership to address zero");
356      emit OwnershipTransferred(owner, _newOwner);
357      owner = _newOwner;
358  }

```

✓ The code meets the specification

Formal Verification Request 13

pause

📅 09, Jun 2019

🕒 3.41 ms

Line 374-379 in File USDKimpl.sol

```
374  /*@CTK pause
375      @tag assume_completion
376      @post owner == msg.sender
377      @post !paused
378      @post __post.paused
379  */
```

Line 380-384 in File USDKimpl.sol

```
380  function pause() public onlyOwner {
381      require(!paused, "already paused");
382      paused = true;
383      emit Pause();
384  }
```

✓ The code meets the specification

Formal Verification Request 14

unpause

📅 09, Jun 2019

🕒 24.61 ms

Line 389-394 in File USDKimpl.sol

```
389  /*@CTK unpause
390      @tag assume_completion
391      @post owner == msg.sender
392      @post paused
393      @post !__post.paused
394  */
```

Line 395-399 in File USDKimpl.sol

```
395  function unpause() public onlyOwner {
396      require(paused, "already unpaused");
397      paused = false;
398      emit Unpause();
399  }
```

✓ The code meets the specification

Formal Verification Request 15

setLawEnforcementRole

📅 09, Jun 2019

🕒 21.22 ms

Line 407-411 in File USDKimpl.sol

```
407  /*@CTK setLawEnforcementRole
408      @tag assume_completion
409      @post msg.sender == lawEnforcementRole || msg.sender == owner
410      @post __post.lawEnforcementRole == _newLawEnforcementRole
411  */
```

Line 412-416 in File USDKimpl.sol

```
412     function setLawEnforcementRole(address _newLawEnforcementRole) public {
413         require(msg.sender == lawEnforcementRole || msg.sender == owner, "only
            lawEnforcementRole or Owner");
414         emit LawEnforcementRoleSet(lawEnforcementRole, _newLawEnforcementRole);
415         lawEnforcementRole = _newLawEnforcementRole;
416     }
```

✓ The code meets the specification

Formal Verification Request 16

freeze

📅 09, Jun 2019

🕒 30.32 ms

Line 427-432 in File USDKimpl.sol

```
427     /*@CTK freeze
428         @tag assume_completion
429         @post msg.sender == lawEnforcementRole
430         @post !frozen[_addr]
431         @post __post.frozen[_addr]
432     */
```

Line 433-437 in File USDKimpl.sol

```
433     function freeze(address _addr) public onlyLawEnforcementRole {
434         require(!frozen[_addr], "address already frozen");
435         frozen[_addr] = true;
436         emit AddressFrozen(_addr);
437     }
```

✓ The code meets the specification

Formal Verification Request 17

unfreeze

📅 09, Jun 2019

🕒 29.54 ms

Line 443-448 in File USDKimpl.sol

```
443     /*@CTK unfreeze
444         @tag assume_completion
445         @post msg.sender == lawEnforcementRole
446         @post frozen[_addr]
447         @post !__post.frozen[_addr]
448     */
```

Line 449-453 in File USDKimpl.sol

```

449     function unfreeze(address _addr) public onlyLawEnforcementRole {
450         require(frozen[_addr], "address already unfrozen");
451         frozen[_addr] = false;
452         emit AddressUnfrozen(_addr);
453     }

```

✓ The code meets the specification

Formal Verification Request 18

wipeFrozenAddress

09, Jun 2019

111.72 ms

Line 460-466 in File USDKimpl.sol

```

460     /*@CTK wipeFrozenAddress
461         @tag assume_completion
462         @post frozen[_addr]
463         @post msg.sender == lawEnforcementRole
464         @post __post.balances[_addr] == 0
465         @post __post.totalSupply_ == totalSupply_ - balances[_addr]
466     */

```

Line 467-475 in File USDKimpl.sol

```

467     function wipeFrozenAddress(address _addr) public onlyLawEnforcementRole {
468         require(frozen[_addr], "address is not frozen");
469         uint256 _balance = balances[_addr];
470         balances[_addr] = 0;
471         totalSupply_ = totalSupply_.sub(_balance);
472         emit FrozenAddressWiped(_addr);
473         emit SupplyDecreased(_addr, _balance);
474         emit Transfer(_addr, address(0), _balance);
475     }

```

✓ The code meets the specification

Formal Verification Request 19

isFrozen

09, Jun 2019

5.55 ms

Line 482-484 in File USDKimpl.sol

```

482     /*@CTK isFrozen
483         @post __return == frozen[_addr]
484     */

```

Line 485-487 in File USDKimpl.sol

```

485     function isFrozen(address _addr) public view returns (bool) {
486         return frozen[_addr];
487     }

```

✓ The code meets the specification

Formal Verification Request 20

setSupplyController

📅 09, Jun 2019

🕒 30.07 ms

Line 495-500 in File USDKimpl.sol

```
495  /*@CTK setSupplyController
496      @tag assume_completion
497      @post msg.sender == supplyController || msg.sender == owner
498      @post _newSupplyController != address(0)
499      @post __post.supplyController == _newSupplyController
500  */
```

Line 501-506 in File USDKimpl.sol

```
501  function setSupplyController(address _newSupplyController) public {
502      require(msg.sender == supplyController || msg.sender == owner, "only
        SupplyController or Owner");
503      require(_newSupplyController != address(0), "cannot set supply controller to
        address zero");
504      emit SupplyControllerSet(supplyController, _newSupplyController);
505      supplyController = _newSupplyController;
506  }
```

✓ The code meets the specification

Formal Verification Request 21

increaseSupply

📅 09, Jun 2019

🕒 110.13 ms

Line 518-523 in File USDKimpl.sol

```
518  /*@CTK increaseSupply
519      @tag assume_completion
520      @post msg.sender == supplyController
521      @post __post.totalSupply_ == totalSupply_ + _value
522      @post __post.balances[supplyController] == balances[supplyController] + _value
523  */
```

Line 524-530 in File USDKimpl.sol

```
524  function increaseSupply(uint256 _value) public onlySupplyController returns (bool
        success) {
525      totalSupply_ = totalSupply_.add(_value);
526      balances[supplyController] = balances[supplyController].add(_value);
527      emit SupplyIncreased(supplyController, _value);
528      emit Transfer(address(0), supplyController, _value);
529      return true;
530  }
```


✓ The code meets the specification

Formal Verification Request 22

decreaseSupply

📅 09, Jun 2019

🕒 206.1 ms

Line 537-542 in File USDKimpl.sol

```
537  /*@CTK decreaseSupply
538      @tag assume_completion
539      @post msg.sender == supplyController
540      @post __post.totalSupply_ == totalSupply_ - _value
541      @post __post.balances[supplyController] == balances[supplyController] - _value
542  */
```

Line 543-550 in File USDKimpl.sol

```
543  function decreaseSupply(uint256 _value) public onlySupplyController returns (bool
      success) {
544      require(_value <= balances[supplyController], "not enough supply");
545      balances[supplyController] = balances[supplyController].sub(_value);
546      totalSupply_ = totalSupply_.sub(_value);
547      emit SupplyDecreased(supplyController, _value);
548      emit Transfer(supplyController, address(0), _value);
549      return true;
550  }
```

✓ The code meets the specification

Formal Verification Request 23

SafeMath sub

📅 09, Jun 2019

🕒 30.45 ms

Line 12-16 in File SafeMath.sol

```
12  /*@CTK "SafeMath sub"
13      @post (a < b) == __reverted
14      @post !__reverted -> __return == a - b
15      @post !__reverted -> !__has_overflow
16  */
```

Line 17-22 in File SafeMath.sol

```
17  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
18      require(b <= a);
19      uint256 c = a - b;
20
21      return c;
22  }
```

✓ The code meets the specification

Formal Verification Request 24

SafeMath add

📅 09, Jun 2019

🕒 15.63 ms

Line 27-31 in File SafeMath.sol

```
27  /*@CTK "SafeMath add"
28     @post (a + b < a || a + b < b) == __reverted
29     @post !__reverted -> __return == a + b
30     @post !__reverted -> !__has_overflow
31  */
```

Line 32-37 in File SafeMath.sol

```
32  function add(uint256 a, uint256 b) internal pure returns (uint256) {
33      uint256 c = a + b;
34      require(c >= a);
35
36      return c;
37  }
```

✅ The code meets the specification

Formal Verification Request 25

Migrations

📅 09, Jun 2019

🕒 9.73 ms

Line 11-13 in File Migrations.sol

```
11  /*@CTK Migrations
12     @post __post.owner == msg.sender
13  */
```

Line 14-16 in File Migrations.sol

```
14  function Migrations() public {
15      owner = msg.sender;
16  }
```

✅ The code meets the specification

Formal Verification Request 26

setCompleted

📅 09, Jun 2019

🕒 10.09 ms

Line 18-21 in File Migrations.sol

```
18  /*@CTK setCompleted
19      @pre msg.sender == owner
20      @post __post.last_completed_migration == completed
21  */
```

Line 22-24 in File Migrations.sol

```
22  function setCompleted(uint completed) public restricted {
23      last_completed_migration = completed;
24  }
```

✓ The code meets the specification

Source Code with CertiK Labels

File USDKimpl.sol

```

1  pragma solidity ^0.4.24;
2
3
4  import "./SafeMath.sol";
5
6
7
8  /**
9   * @title USDKImplementation
10  * @dev this contract is a Pausable ERC20 token with Burn and Mint
11  * controleld by a central SupplyController. By implementing USDKImplementation
12  * this contract also includes external methods for setting
13  * a new implementation contract for the Proxy.
14  * NOTE: The storage defined here will actually be held in the Proxy
15  * contract and all calls to this contract should be made through
16  * the proxy, including admin actions done as owner or supplyController.
17  * Any call to transfer against this contract should fail
18  * with insufficient funds since no tokens will be issued there.
19  */
20 contract USDKImplementation {
21
22     /**
23      * MATH
24      */
25
26     using SafeMath for uint256;
27
28     /**
29      * DATA
30      */
31
32     // INITIALIZATION DATA
33     bool private initialized = false;
34
35     // ERC20 BASIC DATA
36     mapping(address => uint256) internal balances;
37     uint256 internal totalSupply_;
38     string public constant name = "USDK"; // solium-disable-line uppercase
39     string public constant symbol = "USDK"; // solium-disable-line uppercase
40     uint8 public constant decimals = 18; // solium-disable-line uppercase
41
42     // ERC20 DATA
43     mapping (address => mapping (address => uint256)) internal _allowed;
44
45     // OWNER DATA
46     address public owner;
47
48     // PAUSABILITY DATA
49     bool public paused = false;
50
51     // LAW ENFORCEMENT DATA
52     address public lawEnforcementRole;
53     mapping(address => bool) internal frozen;
54

```

```

55 // SUPPLY CONTROL DATA
56
57
58
59 address public supplyController;
60
61 /**
62  * EVENTS
63  */
64
65 // ERC20 BASIC EVENTS
66 event Transfer(address indexed from, address indexed to, uint256 value);
67
68 // ERC20 EVENTS
69 event Approval(
70     address indexed owner,
71     address indexed spender,
72     uint256 value
73 );
74
75 // OWNABLE EVENTS
76 event OwnershipTransferred(
77     address indexed oldOwner,
78     address indexed newOwner
79 );
80
81 // PAUSABLE EVENTS
82 event Pause();
83 event Unpause();
84
85 // LAW ENFORCEMENT EVENTS
86 event AddressFrozen(address indexed addr);
87 event AddressUnfrozen(address indexed addr);
88 event FrozenAddressWiped(address indexed addr);
89 event LawEnforcementRoleSet (
90     address indexed oldLawEnforcementRole,
91     address indexed newLawEnforcementRole
92 );
93
94 // SUPPLY CONTROL EVENTS
95 event SupplyIncreased(address indexed to, uint256 value);
96 event SupplyDecreased(address indexed from, uint256 value);
97 event SupplyControllerSet(
98     address indexed oldSupplyController,
99     address indexed newSupplyController
100 );
101
102 /**
103  * FUNCTIONALITY
104  */
105
106 // INITIALIZATION FUNCTIONALITY
107
108 /**
109  * @dev sets 0 initials tokens, the owner, and the supplyController.
110  * this serves as the constructor for the proxy but compiles to the
111  * memory model of the Implementation contract.
112  */

```

```

113  /*@CTK initialize
114      @tag assume_completion
115      @post !initialized
116      @post __post.initialized
117      @post __post.owner == msg.sender
118      @post __post.lawEnforcementRole == address(0)
119      @post __post.totalSupply_ == 0
120      @post __post.supplyController == msg.sender
121  */
122  function initialize() public {
123      require(!initialized, "already initialized");
124      owner = msg.sender;
125      lawEnforcementRole = address(0);
126      totalSupply_ = 0;
127      supplyController = msg.sender;
128      initialized = true;
129  }
130
131  /**
132   * The constructor is used here to ensure that the implementation
133   * contract is initialized. An uncontrolled implementation
134   * contract might lead to misleading state
135   * for users who accidentally interact with it.
136   */
137  /*@CTK OKUSDImplementation
138      @tag assume_completion
139      @post !initialized && !paused
140      @post __post.initialized && __post.paused
141      @post __post.lawEnforcementRole == address(0)
142      @post __post.totalSupply_ == 0
143      @post __post.supplyController == msg.sender
144  */
145  constructor() public {
146      initialize();
147      pause();
148  }
149
150  // ERC20 BASIC FUNCTIONALITY
151
152  /**
153   * @dev Total number of tokens in existence
154   */
155  /*@CTK totalSupply
156      @post __return == totalSupply_
157  */
158  function totalSupply() public view returns (uint256) {
159      return totalSupply_;
160  }
161
162  /**
163   * @dev Transfer token for a specified address
164   * @param _to The address to transfer to.
165   * @param _value The amount to be transferred.
166   */
167  /*@CTK transfer
168      @tag assume_completion
169      @pre msg.sender != _to
170      @post !paused

```

```

171     @post _to != address(0)
172     @post !frozen[_to] && !frozen[msg.sender]
173     @post _value <= balances[msg.sender]
174     @post __post.balances[msg.sender] == balances[msg.sender] - _value
175     @post __post.balances[_to] == balances[_to] + _value
176     */
177     function transfer(address _to, uint256 _value) public whenNotPaused returns (bool)
178     {
179         require(_to != address(0), "cannot transfer to address zero");
180         require(!frozen[_to] && !frozen[msg.sender], "address frozen");
181         require(_value <= balances[msg.sender], "insufficient funds");
182
183         balances[msg.sender] = balances[msg.sender].sub(_value);
184         balances[_to] = balances[_to].add(_value);
185         emit Transfer(msg.sender, _to, _value);
186         return true;
187     }
188
189     /**
190     * @dev Gets the balance of the specified address.
191     * @param _addr The address to query the the balance of.
192     * @return An uint256 representing the amount owned by the passed address.
193     */
194     /*@CTK balanceOf
195     @post __return == __post.balances[_addr]
196     */
197     function balanceOf(address _addr) public view returns (uint256) {
198         return balances[_addr];
199     }
200
201     // ERC20 FUNCTIONALITY
202
203     /**
204     * @dev Transfer tokens from one address to another
205     * @param _from address The address which you want to send tokens from
206     * @param _to address The address which you want to transfer to
207     * @param _value uint256 the amount of tokens to be transferred
208     */
209     /*@CTK transferFrom
210     @tag assume_completion
211     @pre _to != _from && _to != msg.sender && _from != msg.sender
212     @post !paused
213     @post !frozen[_to] && !frozen[msg.sender] && !frozen[_from]
214     @post _value <= balances[_from]
215     @post _value <= _allowed[_from][msg.sender]
216     @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
217     @post __post.balances[_to] == balances[_to] + _value
218     @post __post.balances[_from] == balances[_from] - _value
219     */
220     function transferFrom(address _from, address _to, uint256 _value) public
221     whenNotPaused returns (bool)
222     {
223         require(_to != address(0), "cannot transfer to address zero");
224         require(!frozen[_to] && !frozen[_from] && !frozen[msg.sender], "address frozen");
225         require(_value <= balances[_from], "insufficient funds");
226         require(_value <= _allowed[_from][msg.sender], "insufficient allowance");

```

```

226     balances[_from] = balances[_from].sub(_value);
227     balances[_to] = balances[_to].add(_value);
228     _allowed[_from][msg.sender] = _allowed[_from][msg.sender].sub(_value);
229     emit Transfer(_from, _to, _value);
230     return true;
231 }
232
233 /**
234  * @dev Approve the passed address to spend the specified amount of tokens on
235  * behalf of msg.sender.
236  * Beware that changing an allowance with this method brings the risk that someone
237  * may use both the old
238  * and the new allowance by unfortunate transaction ordering. One possible
239  * solution to mitigate this
240  * race condition is to first reduce the spender's allowance to 0 and set the
241  * desired value afterwards:
242  * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
243  * @param _spender The address which will spend the funds.
244  * @param _value The amount of tokens to be spent.
245  */
246 /*@CTK approve
247  @tag assume_completion
248  @post !paused
249  @post !frozen[_spender] && !frozen[msg.sender]
250  @post __post._allowed[msg.sender][_spender] == _value
251  */
252 function approve(address _spender, uint256 _value) public whenNotPaused returns (
253     bool) {
254     require(!frozen[_spender] && !frozen[msg.sender], "address frozen");
255     _allowed[msg.sender][_spender] = _value;
256     emit Approval(msg.sender, _spender, _value);
257     return true;
258 }
259
260 /**
261  * @dev Approve an address to spend another addresses' tokens.
262  * @param _owner The address that owns the tokens.
263  * @param spender The address that will spend the tokens.
264  * @param value The number of tokens that can be spent.
265  */
266 /*@CTK _approve
267  @tag assume_completion
268  @post !frozen[spender] && !frozen[_owner]
269  @post spender != address(0) && _owner != address(0)
270  @post __post._allowed[_owner][spender] == value
271  */
272 function _approve(address _owner, address spender, uint256 value) internal {
273     require(!frozen[spender] && !frozen[_owner], "address frozen");
274     require(spender != address(0) && _owner != address(0), "not address(0)");
275     _allowed[_owner][spender] = value;
276     emit Approval(_owner, spender, value);
277 }
278
279 /**
280  * @dev Function to check the amount of tokens that an owner allowed to a spender.
281  * @param _owner address The address which owns the funds.
282  * @param _spender address The address which will spend the funds.

```



```

278     * @return A uint256 specifying the amount of tokens still available for the
        spender.
279     */
280     /*@CTK allowance
281         @post __return == _allowed[_owner][_spender]
282     */
283     function allowance(address _owner, address _spender) public view returns (uint256)
284     {
285         return _allowed[_owner][_spender];
286     }
287
288     /**
289     * @dev Increase the amount of tokens that an owner allowed to a spender.
290     * approve should be called when _allowed[msg.sender][spender] == 0. To increment
291     * allowed value is better to use this function to avoid 2 calls (and wait until
292     * the first transaction is mined)
293     * From MonolithDAO Token.sol
294     * Emits an Approval event.
295     * @param spender The address which will spend the funds.
296     * @param addedValue The amount of tokens to increase the allowance by.
297     */
298     /*@CTK increaseAllowance
299         @tag assume_completion
300         @post !paused
301         @post !frozen[spender] && !frozen[msg.sender]
302         @post spender != address(0) && msg.sender != address(0)
303         @post __post._allowed[msg.sender][spender] ==
304             _allowed[msg.sender][spender] + addedValue
305     */
306     function increaseAllowance(address spender, uint256 addedValue) public
        whenNotPaused returns (bool) {
307         _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
308         return true;
309     }
310
311     /**
312     * @dev Decrease the amount of tokens that an owner allowed to a spender.
313     * approve should be called when _allowed[msg.sender][spender] == 0. To decrement
314     * allowed value is better to use this function to avoid 2 calls (and wait until
315     * the first transaction is mined)
316     * From MonolithDAO Token.sol
317     * Emits an Approval event.
318     * @param spender The address which will spend the funds.
319     * @param subtractedValue The amount of tokens to decrease the allowance by.
320     */
321     /*@CTK decreaseAllowance
322         @tag assume_completion
323         @post !paused
324         @post !frozen[spender] && !frozen[msg.sender]
325         @post spender != address(0) && msg.sender != address(0)
326         @post __post._allowed[msg.sender][spender] ==
327             _allowed[msg.sender][spender] - subtractedValue
328     */
329     function decreaseAllowance(address spender, uint256 subtractedValue) public
        whenNotPaused returns (bool) {
330         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
        ));
331         return true;

```

```
332 }
333
334 // OWNER FUNCTIONALITY
335
336 /**
337  * @dev Throws if called by any account other than the owner.
338  */
339 modifier onlyOwner() {
340     require(msg.sender == owner, "onlyOwner");
341     _;
342 }
343
344 /**
345  * @dev Allows the current owner to transfer control of the contract to a newOwner
346  * @param _newOwner The address to transfer ownership to.
347  */
348 /*@CTK transferOwnership
349  @tag assume_completion
350  @post owner == msg.sender
351  @post _newOwner != address(0)
352  @post __post.owner == _newOwner
353  */
354 function transferOwnership(address _newOwner) public onlyOwner {
355     require(_newOwner != address(0), "cannot transfer ownership to address zero");
356     emit OwnershipTransferred(owner, _newOwner);
357     owner = _newOwner;
358 }
359
360 // PAUSABILITY FUNCTIONALITY
361
362 /**
363  * @dev Modifier to make a function callable only when the contract is not paused.
364  */
365 modifier whenNotPaused() {
366     require(!paused, "whenNotPaused");
367     _;
368 }
369
370 /**
371  * @dev called by the owner to pause, triggers stopped state
372  */
373 /*@CTK pause
374  @tag assume_completion
375  @post owner == msg.sender
376  @post !paused
377  @post __post.paused
378  */
379 function pause() public onlyOwner {
380     require(!paused, "already paused");
381     paused = true;
382     emit Pause();
383 }
384
385 /**
386  * @dev called by the owner to unpause, returns to normal state
387  */
```

```

389  /*@CTK unpause
390      @tag assume_completion
391      @post owner == msg.sender
392      @post paused
393      @post !__post.paused
394  */
395  function unpause() public onlyOwner {
396      require(paused, "already unpaused");
397      paused = false;
398      emit Unpause();
399  }
400
401  // LAW ENFORCEMENT FUNCTIONALITY
402
403  /**
404   * @dev Sets a new law enforcement role address.
405   * @param _newLawEnforcementRole The new address allowed to freeze/unfreeze
406   *   addresses and seize their tokens.
407  */
408  /*@CTK setLawEnforcementRole
409      @tag assume_completion
410      @post msg.sender == lawEnforcementRole || msg.sender == owner
411      @post __post.lawEnforcementRole == _newLawEnforcementRole
412  */
413  function setLawEnforcementRole(address _newLawEnforcementRole) public {
414      require(msg.sender == lawEnforcementRole || msg.sender == owner, "only
415      lawEnforcementRole or Owner");
416      emit LawEnforcementRoleSet(lawEnforcementRole, _newLawEnforcementRole);
417      lawEnforcementRole = _newLawEnforcementRole;
418  }
419
420  modifier onlyLawEnforcementRole() {
421      require(msg.sender == lawEnforcementRole, "onlyLawEnforcementRole");
422      _;
423  }
424
425  /**
426   * @dev Freezes an address balance from being transferred.
427   * @param _addr The new address to freeze.
428  */
429  /*@CTK freeze
430      @tag assume_completion
431      @post msg.sender == lawEnforcementRole
432      @post !frozen[_addr]
433      @post __post.frozen[_addr]
434  */
435  function freeze(address _addr) public onlyLawEnforcementRole {
436      require(!frozen[_addr], "address already frozen");
437      frozen[_addr] = true;
438      emit AddressFrozen(_addr);
439  }
440
441  /**
442   * @dev Unfreezes an address balance allowing transfer.
443   * @param _addr The new address to unfreeze.
444  */
445  /*@CTK unfreeze
446      @tag assume_completion

```

```

445     @post msg.sender == lawEnforcementRole
446     @post frozen[_addr]
447     @post !__post.frozen[_addr]
448     */
449     function unfreeze(address _addr) public onlyLawEnforcementRole {
450         require(frozen[_addr], "address already unfrozen");
451         frozen[_addr] = false;
452         emit AddressUnfrozen(_addr);
453     }
454
455     /**
456     * @dev Wipes the balance of a frozen address, burning the tokens
457     * and setting the approval to zero.
458     * @param _addr The new frozen address to wipe.
459     */
460     /*@CTK wipeFrozenAddress
461     @tag assume_completion
462     @post frozen[_addr]
463     @post msg.sender == lawEnforcementRole
464     @post __post.balances[_addr] == 0
465     @post __post.totalSupply_ == totalSupply_ - balances[_addr]
466     */
467     function wipeFrozenAddress(address _addr) public onlyLawEnforcementRole {
468         require(frozen[_addr], "address is not frozen");
469         uint256 _balance = balances[_addr];
470         balances[_addr] = 0;
471         totalSupply_ = totalSupply_.sub(_balance);
472         emit FrozenAddressWiped(_addr);
473         emit SupplyDecreased(_addr, _balance);
474         emit Transfer(_addr, address(0), _balance);
475     }
476
477     /**
478     * @dev Gets the balance of the specified address.
479     * @param _addr The address to check if frozen.
480     * @return A bool representing whether the given address is frozen.
481     */
482     /*@CTK isFrozen
483     @post __return == frozen[_addr]
484     */
485     function isFrozen(address _addr) public view returns (bool) {
486         return frozen[_addr];
487     }
488
489     // SUPPLY CONTROL FUNCTIONALITY
490
491     /**
492     * @dev Sets a new supply controller address.
493     * @param _newSupplyController The address allowed to burn/mint tokens to control
494     * supply.
495     */
496     /*@CTK setSupplyController
497     @tag assume_completion
498     @post msg.sender == supplyController || msg.sender == owner
499     @post _newSupplyController != address(0)
500     @post __post.supplyController == _newSupplyController
501     */
502     function setSupplyController(address _newSupplyController) public {

```

```

502     require(msg.sender == supplyController || msg.sender == owner, "only
503         SupplyController or Owner");
504     require(_newSupplyController != address(0), "cannot set supply controller to
505         address zero");
506     emit SupplyControllerSet(supplyController, _newSupplyController);
507     supplyController = _newSupplyController;
508 }
509
510 modifier onlySupplyController() {
511     require(msg.sender == supplyController, "onlySupplyController");
512     _;
513 }
514
515 /**
516  * @dev Increases the total supply by minting the specified number of tokens to
517  * the supply controller account.
518  * @param _value The number of tokens to add.
519  * @return A boolean that indicates if the operation was successful.
520  */
521 /*@CTK increaseSupply
522  @tag assume_completion
523  @post msg.sender == supplyController
524  @post __post.totalSupply_ == totalSupply_ + _value
525  @post __post.balances[supplyController] == balances[supplyController] + _value
526  */
527 function increaseSupply(uint256 _value) public onlySupplyController returns (bool
528     success) {
529     totalSupply_ = totalSupply_.add(_value);
530     balances[supplyController] = balances[supplyController].add(_value);
531     emit SupplyIncreased(supplyController, _value);
532     emit Transfer(address(0), supplyController, _value);
533     return true;
534 }
535
536 /**
537  * @dev Decreases the total supply by burning the specified number of tokens from
538  * the supply controller account.
539  * @param _value The number of tokens to remove.
540  * @return A boolean that indicates if the operation was successful.
541  */
542 /*@CTK decreaseSupply
543  @tag assume_completion
544  @post msg.sender == supplyController
545  @post __post.totalSupply_ == totalSupply_ - _value
546  @post __post.balances[supplyController] == balances[supplyController] - _value
547  */
548 function decreaseSupply(uint256 _value) public onlySupplyController returns (bool
549     success) {
550     require(_value <= balances[supplyController], "not enough supply");
551     balances[supplyController] = balances[supplyController].sub(_value);
552     totalSupply_ = totalSupply_.sub(_value);
553     emit SupplyDecreased(supplyController, _value);
554     emit Transfer(supplyController, address(0), _value);
555     return true;
556 }
557 }

```

File SafeMath.sol

```

1 pragma solidity ^0.4.0;
2
3
4 /**
5  * @title SafeMath
6  * @dev Math operations with safety checks that throw on error
7  */
8 library SafeMath {
9     /**
10     * @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater
11         than minuend).
12     */
13     /*@CTK "SafeMath sub"
14     @post (a < b) == __reverted
15     @post !__reverted -> __return == a - b
16     @post !__reverted -> !__has_overflow
17     */
18     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
19         require(b <= a);
20         uint256 c = a - b;
21
22         return c;
23     }
24
25     /**
26     * @dev Adds two numbers, reverts on overflow.
27     */
28     /*@CTK "SafeMath add"
29     @post (a + b < a || a + b < b) == __reverted
30     @post !__reverted -> __return == a + b
31     @post !__reverted -> !__has_overflow
32     */
33     function add(uint256 a, uint256 b) internal pure returns (uint256) {
34         uint256 c = a + b;
35         require(c >= a);
36
37         return c;
38     }
39 }

```

File Migrations.sol

```

1 pragma solidity ^0.4.17;
2
3 contract Migrations {
4     address public owner;
5     uint public last_completed_migration;
6
7     modifier restricted() {
8         if (msg.sender == owner) _;
9     }
10
11     /*@CTK Migrations
12     @post __post.owner == msg.sender
13     */
14     function Migrations() public {
15         owner = msg.sender;
16     }
17 }

```

```
18  /*@CTK setCompleted
19      @pre msg.sender == owner
20      @post __post.last_completed_migration == completed
21  */
22  function setCompleted(uint completed) public restricted {
23      last_completed_migration = completed;
24  }
25
26  function upgrade(address new_address) public restricted {
27      Migrations upgraded = Migrations(new_address);
28      upgraded.setCompleted(last_completed_migration);
29  }
30  }
```