

CERTiK VERIFICATION REPORT FOR BODHI



Bodhi 菩提

Request Date: 2019-03-01

Revision Date: 2019-03-03

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 Bodhi(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.

PASS

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

Mar 03, 2019



Summary

This audit report summarises the smart contract verification service requested by Bodhi. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security loopholes.

The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the time of the audit.

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.	4	SWC-116

Insecure Compiler Version	Com-	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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	Ran-	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	for	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	to	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Variable	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Default	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

No issue found.

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.

Source Code with CertiK Labels

File BaseContract.sol

```

1  pragma solidity ^0.4.18;
2
3
4  contract BaseContract {
5      struct ResultBalance {
6          uint256 totalBets;
7          uint256 totalVotes;
8          mapping(address => uint256) bets;
9          mapping(address => uint256) votes;
10     }
11
12     uint8 public constant INVALID_RESULT_INDEX = 255;
13
14     uint8 public numOfResults;
15     uint8 public resultIndex = INVALID_RESULT_INDEX;
16     uint16 public version;
17     ResultBalance[11] internal balances;
18
19     // Modifiers
20     modifier validResultIndex(uint8 _resultIndex) {
21         require (_resultIndex <= numOfResults - 1);
22         _;
23     }
24
25     /*
26     * @notice Gets the bet balances of the sender for all the results.
27     * @return An array of all the bet balances of the sender.
28     */
29     /*@CTK get_bet_balances
30     @tag spec
31     @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j]
32         .bets[msg.sender]
33     */
34     function getBetBalances()
35     public
36     view
37     returns (uint256[11])
38     {
39         uint256[11] memory betBalances;
40         /*@CTK set_bet_balances
41         @var uint8 i
42         @var BaseContract this
43         @var uint256[11] betBalances
44         @inv forall j: uint. (j >= 0 /\ j < i) -> betBalances[j] == this.balances[j].
45             bets[msg.sender]
46         @inv i <= this.numOfResults
47         @inv this == this__pre
48         @post i >= numOfResults
49         @post !__should_return
50         */
51         for (uint8 i = 0; i < numOfResults; i++) {
52             betBalances[i] = balances[i].bets[msg.sender];
53         }
54         return betBalances;
55     }
56 }

```

```

53     }
54
55     /*
56     * @notice Gets total bets for all the results.
57     * @return An array of total bets for all results.
58     */
59     /*@CTK get_total_bets
60     @tag spec
61     @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j]
        .totalBets
62     */
63     function getTotalBets()
64     public
65     view
66     returns (uint256[11])
67     {
68         uint256[11] memory totalBets;
69         /*@CTK set_total_bets
70         @var uint8 i
71         @var BaseContract this
72         @var uint256[11] totalBets
73         @inv forall j: uint. (j >= 0 /\ j < i) -> totalBets[j] == this.balances[j].
            totalBets
74         @inv i <= this.numOfResults
75         @inv this == this__pre
76         @post i >= numOfResults
77         @post !__should_return
78         */
79         for (uint8 i = 0; i < numOfResults; i++) {
80             totalBets[i] = balances[i].totalBets;
81         }
82         return totalBets;
83     }
84
85     /*
86     * @notice Gets the vote balances of the sender for all the results.
87     * @return An array of all the vote balances of the sender.
88     */
89     /*@CTK get_vote_balances
90     @tag spec
91     @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j]
        .votes[msg.sender]
92     */
93     function getVoteBalances()
94     public
95     view
96     returns (uint256[11])
97     {
98         uint256[11] memory voteBalances;
99         /*@CTK set_vote_balances
100        @var uint8 i
101        @var BaseContract this
102        @var uint256[11] voteBalances
103        @inv forall j: uint. (j >= 0 /\ j < i) -> voteBalances[j] == this.balances[j]
            .votes[msg.sender]
104        @inv i <= this.numOfResults
105        @inv this == this__pre
106        @post i >= numOfResults

```

```

107     @post !__should_return
108     */
109     for (uint8 i = 0; i < numOfResults; i++) {
110         voteBalances[i] = balances[i].votes[msg.sender];
111     }
112     return voteBalances;
113 }
114
115 /*
116 * @notice Gets total votes for all the results.
117 * @return An array of total votes for all results.
118 */
119 /*@CTK get_total_votes
120 @tag spec
121 @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j
122     ].totalVotes
123 */
124 function getTotalVotes()
125 public
126 view
127 returns (uint256[11])
128 {
129     uint256[11] memory totalVotes;
130     /*@CTK set_total_votes
131     @var uint8 i
132     @var BaseContract this
133     @var uint256[11] totalVotes
134     @inv forall j: uint. (j >= 0 /\ j < i) -> totalVotes[j] == this.balances[j].
135         totalVotes
136     @inv i <= this.numOfResults
137     @inv this == this__pre
138     @post i >= numOfResults
139     @post !__should_return
140     */
141     for (uint8 i = 0; i < numOfResults; i++) {
142         totalVotes[i] = balances[i].totalVotes;
143     }
144     return totalVotes;
145 }
146 }

```

File Migrations.sol

```

1 pragma solidity ^0.4.15;
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 init_migrations
12     @post __post.owner == msg.sender
13     */
14     function Migrations() public {
15         owner = msg.sender;
16     }

```



```

17
18  /*@CTK set_complete
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  //@CTK NO_ASF
27  function upgrade(address new_address) public restricted {
28      Migrations upgraded = Migrations(new_address);
29      upgraded.setCompleted(last_completed_migration);
30  }
31  }

```

File mocks/StandardTokenMock.sol

```

1  pragma solidity ^0.4.18;
2
3  import '../tokens/StandardToken.sol';
4
5  contract StandardTokenMock is StandardToken {
6      /*@CTK init_mock_standard_token
7         @post __post.balances[_initialAccount] == _initialBalance
8         @post __post.totalSupply == _initialBalance
9      */
10     function StandardTokenMock(address _initialAccount,
11                                uint256 _initialBalance) public {
12         balances[_initialAccount] = _initialBalance;
13         totalSupply = _initialBalance;
14     }
15 }

```

File mocks/BasicTokenMock.sol

```

1  pragma solidity ^0.4.18;
2
3  import '../tokens/BasicToken.sol';
4
5  contract BasicTokenMock is BasicToken {
6      /*@CTK init_mock_basic_token
7         @post __post.balances[_initialAccount] == _initialBalance
8         @post __post.totalSupply == _initialBalance
9      */
10     function BasicTokenMock(address _initialAccount, uint256 _initialBalance) public {
11         balances[_initialAccount] = _initialBalance;
12         totalSupply = _initialBalance;
13     }
14 }

```

File oracles/CentralizedOracle.sol

```

1  pragma solidity ^0.4.18;
2
3  import "./Oracle.sol";
4
5  contract CentralizedOracle is Oracle {
6      address public oracle;
7      uint256 public bettingStartTime;
8      uint256 public bettingEndTime;

```

```

9      uint256 public resultSettingStartTime;
10     uint256 public resultSettingEndTime;
11
12     /*
13     * @notice Creates new CentralizedOracle contract.
14     * @param _version The contract version.
15     * @param _owner The address of the owner.
16     * @param _eventAddress The address of the Event.
17     * @param _numOfResults The number of result options.
18     * @param _oracle The address of the CentralizedOracle that will ultimately decide
19         the result.
20     * @param _bettingStartTime The unix time when betting will start.
21     * @param _bettingEndTime The unix time when betting will end.
22     * @param _resultSettingStartTime The unix time when the CentralizedOracle can set
23         the result.
24     * @param _resultSettingEndTime The unix time when anyone can set the result.
25     * @param _consensusThreshold The BOT amount that needs to be paid by the Oracle
26         for their result to be valid.
27     */
28     /*@CTK "CentralizedOracle constructor"
29     @tag assume_completion
30     @post __post.eventAddress == _eventAddress
31     @post __post.numOfResults == _numOfResults
32     @post __post.oracle == _oracle
33     @post __post.bettingStartTime == _bettingStartTime
34     @post __post.bettingEndTime == _bettingEndTime
35     @post __post.resultSettingStartTime == _resultSettingStartTime
36     @post __post.resultSettingEndTime == _resultSettingEndTime
37     @post __post.consensusThreshold == _consensusThreshold
38     */
39     /*@CTK "contract failed with invalid input"
40     @pre _numOfResults == 0 \/_ _consensusThreshold == 0
41     @post __reverted == true
42     */
43     function CentralizedOracle(
44         uint16 _version,
45         address _owner,
46         address _eventAddress,
47         uint8 _numOfResults,
48         address _oracle,
49         uint256 _bettingStartTime,
50         uint256 _bettingEndTime,
51         uint256 _resultSettingStartTime,
52         uint256 _resultSettingEndTime,
53         uint256 _consensusThreshold)
54         Ownable(_owner)
55         public
56         validAddress(_oracle)
57         validAddress(_eventAddress)
58     {
59         require(_numOfResults > 0);
60         require(_bettingEndTime > _bettingStartTime);
61         require(_resultSettingStartTime >= _bettingEndTime);
62         require(_resultSettingEndTime > _resultSettingStartTime);
63         require(_consensusThreshold > 0);
64
65         version = _version;
66         eventAddress = _eventAddress;

```

```

64     numOfResults = _numOfResults;
65     oracle = _oracle;
66     bettingStartTime = _bettingStartTime;
67     bettingEndTime = _bettingEndTime;
68     resultSettingStartTime = _resultSettingStartTime;
69     resultSettingEndTime = _resultSettingEndTime;
70     consensusThreshold = _consensusThreshold;
71 }
72
73 /// @notice Fallback function that rejects any amount sent to the contract.
74 function() external payable {
75     revert();
76 }
77
78 /*
79 * @notice Allows betting on a result using the blockchain token.
80 * @param _resultIndex The index of result to bet on.
81 */
82 /*CTK "Bodhi bet against certain index"
83    @tag assume_completion
84    @pre numOfResults > 0
85    @post __post.balances[_resultIndex].totalBets == balances[_resultIndex].
86           totalBets + msg.value
87    @post __post.balances[_resultIndex].bets[msg.sender] == balances[_resultIndex].
88           bets[msg.sender] + msg.value
89    @post __has_overflow == false
90 */
91 function bet(uint8 _resultIndex)
92     external
93     payable
94     validResultIndex(_resultIndex)
95     isNotFinished()
96 {
97     require(block.timestamp >= bettingStartTime);
98     require(block.timestamp < bettingEndTime);
99     require(msg.value > 0);
100
101     balances[_resultIndex].totalBets = balances[_resultIndex].totalBets.add(msg.
102         value);
103     balances[_resultIndex].bets[msg.sender] = balances[_resultIndex].bets[msg.
104         sender].add(msg.value);
105
106     ITopicEvent(eventAddress).betFromOracle.value(msg.value)(msg.sender,
107         _resultIndex);
108
109     OracleResultVoted(version, address(this), msg.sender, _resultIndex, msg.value);
110 }
111
112 /*
113 * @notice CentralizedOracle should call this to set the result. Requires the
114 *       Oracle to approve() BOT in the amount
115 *       of the consensus threshold.
116 * @param _resultIndex The index of the result to set.
117 */
118 /*CTK"Set index result"
119    @tag assume_completion
120    @pre numOfResults > 0
121    @post __post.balances[_resultIndex].totalVotes == balances[_resultIndex].

```

```

116         totalVotes + this.consensusThreshold
117         @post __post.balances[_resultIndex].votes[msg.sender] == balances[_resultIndex].
118         votes[msg.sender] + this.consensusThreshold
119         @post __has_overflow == false
120     */
121     function setResult(uint8 _resultIndex)
122     external
123     validResultIndex(_resultIndex)
124     isNotFinished()
125     {
126         require(block.timestamp >= resultSettingStartTime);
127         if (block.timestamp < resultSettingEndTime) {
128             require(msg.sender == oracle);
129         }
130
131         finished = true;
132         resultIndex = _resultIndex;
133
134         balances[_resultIndex].totalVotes = balances[_resultIndex].totalVotes.add(
135             consensusThreshold);
136         balances[_resultIndex].votes[msg.sender] = balances[_resultIndex].votes[msg.
137             sender].add(consensusThreshold);
138
139         ITopicEvent(eventAddress).centralizedOracleSetResult(msg.sender, _resultIndex,
140             consensusThreshold);
141         OracleResultSet(version, address(this), _resultIndex);
142     }
143 }

```

File oracles/OracleFactory.sol

```

1 pragma solidity ^0.4.18;
2
3 import "./IOracleFactory.sol";
4 import "./CentralizedOracle.sol";
5 import "./DecentralizedOracle.sol";
6 import "./storage/IAddressManager.sol";
7
8 contract OracleFactory is IOracleFactory {
9     uint16 public version;
10     address private addressManager;
11     mapping(bytes32 => address) public oracles;
12
13     // Events
14     event CentralizedOracleCreated(
15         uint16 indexed _version,
16         address indexed _contractAddress,
17         address indexed _eventAddress,
18         uint8 _numOfResults,
19         address _oracle,
20         uint256 _bettingStartTime,
21         uint256 _bettingEndTime,
22         uint256 _resultSettingStartTime,
23         uint256 _resultSettingEndTime,
24         uint256 _consensusThreshold);
25     event DecentralizedOracleCreated(
26         uint16 indexed _version,
27         address indexed _contractAddress,
28         address indexed _eventAddress,

```

```

29     uint8 _numOfResults,
30     uint8 _lastResultIndex,
31     uint256 _arbitrationEndTime,
32     uint256 _consensusThreshold);
33
34     /*
35     * @notice Creates new OracleFactory contract.
36     * @param _addressManager The address of the AddressManager contract.
37     */
38     /*@CTK "OracleFactory constructor"
39     @post __reverted == false -> (__post.addressManager == _addressManager)
40     */
41     function OracleFactory(address _addressManager) public {
42         require(_addressManager != address(0));
43
44         addressManager = _addressManager;
45         // version = IAddressManager(addressManager).currentOracleFactoryIndex();
46     }
47
48     function createCentralizedOracle(
49         address _eventAddress,
50         uint8 _numOfResults,
51         address _oracle,
52         uint256 _bettingStartTime,
53         uint256 _bettingEndTime,
54         uint256 _resultSettingStartTime,
55         uint256 _resultSettingEndTime,
56         uint256 _consensusThreshold)
57     public
58     returns (address)
59     {
60         bytes32 hash = getCentralizedOracleHash(_eventAddress, _numOfResults, _oracle,
61             _bettingStartTime,
62             _bettingEndTime, _resultSettingStartTime, _resultSettingEndTime,
63             _consensusThreshold);
64         // CentralizedOracle should not exist yet
65         require(oracles[hash] == address(0));
66
67         CentralizedOracle cOracle = new CentralizedOracle(version, msg.sender,
68             _eventAddress, _numOfResults, _oracle,
69             _bettingStartTime, _bettingEndTime, _resultSettingStartTime,
70             _resultSettingEndTime, _consensusThreshold);
71         oracles[hash] = address(cOracle);
72
73         CentralizedOracleCreated(version, address(cOracle), _eventAddress,
74             _numOfResults, _oracle, _bettingStartTime,
75             _bettingEndTime, _resultSettingStartTime, _resultSettingEndTime,
76             _consensusThreshold);
77
78         return address(cOracle);
79     }
80
81     function createDecentralizedOracle(
82         address _eventAddress,
83         uint8 _numOfResults,
84         uint8 _lastResultIndex,

```

```

81     uint256 _arbitrationEndTime,
82     uint256 _consensusThreshold)
83     public
84     returns (address)
85     {
86         bytes32 hash = getDecentralizedOracleHash(_eventAddress, _numOfResults,
87             _lastResultIndex, _arbitrationEndTime,
88             _consensusThreshold);
89         // DecentralizedOracle should not exist yet
90         require(oracles[hash] == address(0));
91
92         DecentralizedOracle dOracle = new DecentralizedOracle(version, msg.sender,
93             _eventAddress, _numOfResults,
94             _lastResultIndex, _arbitrationEndTime, _consensusThreshold);
95         oracles[hash] = address(dOracle);
96
97         DecentralizedOracleCreated(version, address(dOracle), _eventAddress,
98             _numOfResults, _lastResultIndex,
99             _arbitrationEndTime, _consensusThreshold);
100
101         return address(dOracle);
102     }
103
104     function getCentralizedOracleHash(
105         address _eventAddress,
106         uint8 _numOfResults,
107         address _oracle,
108         uint256 _bettingStartTime,
109         uint256 _bettingEndTime,
110         uint256 _resultSettingStartTime,
111         uint256 _resultSettingEndTime,
112         uint256 _consensusThreshold)
113         private
114         pure
115         returns (bytes32)
116     {
117         return keccak256(_eventAddress, _numOfResults, _oracle, _bettingStartTime,
118             _bettingEndTime,
119             _resultSettingStartTime, _resultSettingEndTime, _consensusThreshold);
120     }
121
122     function getDecentralizedOracleHash(
123         address _eventAddress,
124         uint8 _numOfResults,
125         uint8 _lastResultIndex,
126         uint256 _arbitrationEndTime,
127         uint256 _consensusThreshold)
128         private
129         pure
130         returns (bytes32)
131     {
132         return keccak256(_eventAddress, _numOfResults, _lastResultIndex,
133             _arbitrationEndTime, _consensusThreshold);
134     }
135 }

```

File oracles/DecentralizedOracle.sol

```

1  pragma solidity ^0.4.18;
2
3  import "./Oracle.sol";
4
5  contract DecentralizedOracle is Oracle {
6      uint8 public lastResultIndex;
7      uint256 public arbitrationEndTime;
8
9      /*
10     * @notice Creates new DecentralizedOracle contract.
11     * @param _version The contract version.
12     * @param _owner The address of the owner.
13     * @param _eventAddress The address of the Event.
14     * @param _numOfResults The number of result options.
15     * @param _lastResultIndex The last result index set by the DecentralizedOracle.
16     * @param _arbitrationEndTime The unix time when the voting period ends.
17     * @param _consensusThreshold The BOT amount that needs to be reached for this
18         DecentralizedOracle to be valid.
19     */
20     /*@CTK "DecentralizedOracle constructor"
21     @tag assume_completion
22     @pre _numOfResults > 0
23     @pre _consensusThreshold > 0
24     @post __post.version == _version
25     @post __post.eventAddress == _eventAddress
26     @post __post.numOfResults == _numOfResults
27     @post __post.lastResultIndex == _lastResultIndex
28     @post __post.arbitrationEndTime == _arbitrationEndTime
29     @post __post.consensusThreshold == _consensusThreshold
30     */
31     /*CTK "DecentralizedOracle construct fail with invalid input"
32     @pre _numOfResults == 0 \\/ _consensusThreshold == 0
33     @post __reverted == true
34     */
35     function DecentralizedOracle(
36         uint16 _version,
37         address _owner,
38         address _eventAddress,
39         uint8 _numOfResults,
40         uint8 _lastResultIndex,
41         uint256 _arbitrationEndTime,
42         uint256 _consensusThreshold)
43         Ownable(_owner)
44         public
45         validAddress(_eventAddress)
46     {
47         require(_numOfResults > 0);
48         require(_arbitrationEndTime > block.timestamp);
49         require(_consensusThreshold > 0);
50
51         version = _version;
52         eventAddress = _eventAddress;
53         numOfResults = _numOfResults;
54         lastResultIndex = _lastResultIndex;
55         arbitrationEndTime = _arbitrationEndTime;
56         consensusThreshold = _consensusThreshold;
57     }

```

```

58  /*
59  * @notice Vote on an Event result which requires BOT payment.
60  * @param _eventResultIndex The Event result which is being voted on.
61  * @param _botAmount The amount of BOT used to vote.
62  */
63  function voteResult(uint8 _eventResultIndex, uint256 _botAmount)
64      external
65      validResultIndex(_eventResultIndex)
66      isNotFinished()
67  {
68      require(_botAmount > 0);
69      require(block.timestamp < arbitrationEndTime);
70      require(_eventResultIndex != lastResultIndex);
71
72      // Only accept the vote amount up to the consensus threshold
73      uint256 adjustedVoteAmount = _botAmount;
74      if (balances[_eventResultIndex].totalVotes.add(_botAmount) > consensusThreshold
75          ) {
76          adjustedVoteAmount = consensusThreshold.sub(balances[_eventResultIndex].
77              totalVotes);
78      }
79
80      balances[_eventResultIndex].totalVotes = balances[_eventResultIndex].totalVotes
81          .add(adjustedVoteAmount);
82      balances[_eventResultIndex].votes[msg.sender] = balances[_eventResultIndex].
83          votes[msg.sender]
84          .add(adjustedVoteAmount);
85
86      ITopicEvent(eventAddress).voteFromOracle(_eventResultIndex, msg.sender,
87          adjustedVoteAmount);
88      OracleResultVoted(version, address(this), msg.sender, _eventResultIndex,
89          adjustedVoteAmount);
90
91      if (balances[_eventResultIndex].totalVotes >= consensusThreshold) {
92          setResult();
93      }
94  }
95
96  /*
97  * @notice This can be called by anyone if this VotingOracle did not meet the
98  * consensus threshold and has reached
99  * the arbitration end time. This finishes the Event and allows winners to
100  * withdraw their winnings from the Event
101  * contract.
102  * @return Flag to indicate success of finalizing the result.
103  */
104  function finalizeResult()
105      external
106      isNotFinished()
107  {
108      require(block.timestamp >= arbitrationEndTime);
109
110      finished = true;
111      resultIndex = lastResultIndex;
112
113      ITopicEvent(eventAddress).decentralizedOracleFinalizeResult();
114  }

```



```

108  /*
109  * @dev DecentralizedOracle is validated and set the result of the Event.
110  */
111  function setResult()
112  private
113  {
114      finished = true;
115
116      uint256 winningVoteBalance = 0;
117      for (uint8 i = 0; i < numOfResults; i++) {
118          uint256 totalVoteBalance = balances[i].totalVotes;
119          if (totalVoteBalance > winningVoteBalance) {
120              winningVoteBalance = totalVoteBalance;
121              resultIndex = i;
122          }
123      }
124
125      ITopicEvent(eventAddress).decentralizedOracleSetResult(resultIndex,
126          winningVoteBalance);
127      OracleResultSet(version, address(this), resultIndex);
128  }

```

File storage/AddressManager.sol

```

1  pragma solidity ^0.4.18;
2
3  import "./IAddressManager.sol";
4  import "../libs/Ownable.sol";
5  import "../tokens/ERC20.sol";
6
7  contract AddressManager is IAddressManager, Ownable {
8      uint256 public constant botDecimals = 8; // Number of decimals for BOT
9
10     uint16 public currentEventFactoryIndex = 0; // Version of the next upgraded
        EventFactory contract
11     uint16 public currentOracleFactoryIndex = 0; // Version of the next upgraded
        OracleFactory contract
12     uint256 public eventEscrowAmount = 100 * (10**botDecimals); // Amount of escrow
        deposit needed to create an event
13     uint256 public arbitrationLength = 86400; // Number of seconds for arbitration
        period
14     uint256 public startingOracleThreshold = 100 * (10**botDecimals); // Consensus
        threshold for CentralizedOracles
15     uint256 public thresholdPercentIncrease = 10; // Percentage to increase the
        Consensus Threshold every round
16     mapping(address => uint16) public eventFactoryAddressToVersion;
17     mapping(address => uint16) public oracleFactoryAddressToVersion;
18     mapping(address => bool) private whitelistedContracts;
19
20     // Events
21     event BodhiTokenAddressChanged(address indexed _newAddress);
22     event EventFactoryAddressAdded(uint16 _index, address indexed _contractAddress);
23     event OracleFactoryAddressAdded(uint16 _index, address indexed _contractAddress);
24     event EscrowDeposited(address indexed _depositor, uint256 escrowAmount);
25     event EscrowWithdrawn(address indexed _eventAddress, address indexed _depositor,
        uint256 escrowAmount);
26     event ContractWhitelisted(address indexed _contractAddress);
27

```

```

28 // Modifiers
29 modifier isWhitelisted(address _contractAddress) {
30     require(whitelistedContracts[_contractAddress] == true);
31     _;
32 }
33
34 function AddressManager() Ownable(msg.sender) public {
35 }
36
37 /*
38 * @notice Transfer the escrow amount needed to create an Event.
39 * @param _creator The address of the creator.
40 */
41 function transferEscrow(address _creator)
42     external
43     isWhitelisted(msg.sender)
44 {
45
46     ERC20 token = ERC20(bodhiTokenAddress);
47     require(token.allowance(_creator, address(this)) >= eventEscrowAmount);
48
49     token.transferFrom(_creator, address(this), eventEscrowAmount);
50
51     EscrowDeposited(_creator, eventEscrowAmount);
52 }
53
54 /*
55 * @notice Withdraws the escrow for an Event.
56 * @param _creator The address of the creator.
57 */
58
59 function withdrawEscrow(address _creator, uint256 _escrowAmount)
60     external
61     isWhitelisted(msg.sender)
62 {
63     ERC20(bodhiTokenAddress).transfer(_creator, _escrowAmount);
64
65     EscrowWithdrawn(msg.sender, _creator, _escrowAmount);
66 }
67
68 /*
69 * @dev Adds a whitelisted contract address. Only allowed to be called from
70 *     previously whitelisted addresses.
71 * @param _contractAddress The address of the contract to whitelist.
72 */
73 /*@CTK whitelist_success
74     @pre whitelistedContracts[_contractAddress] == true
75     @pre _contractAddress != address(0)
76     @post __post.whitelistedContracts[_contractAddress] == true
77 */
78 function addWhitelistContract(address _contractAddress)
79     external
80     isWhitelisted(msg.sender)
81     validAddress(_contractAddress)
82 {
83     whitelistedContracts[_contractAddress] = true;
84
85     ContractWhitelisted(_contractAddress);

```

```

85     }
86
87     /// @dev Allows the owner to set the address of the Bodhi Token contract.
88     /// @param _tokenAddress The address of the Bodhi Token contract.
89     /*@CTK set_bodhi_token_address_success
90         @pre _tokenAddress != address(0)
91         @pre owner == msg.sender
92         @post __post.whitelistedContracts[_tokenAddress] == true
93         @post __post.bodhiTokenAddress == _tokenAddress
94     */
95     function setBodhiTokenAddress(address _tokenAddress)
96     public
97     onlyOwner()
98     validAddress(_tokenAddress)
99     {
100         bodhiTokenAddress = _tokenAddress;
101         whitelistedContracts[_tokenAddress] = true;
102
103         BodhiTokenAddressChanged(bodhiTokenAddress);
104         ContractWhitelisted(_tokenAddress);
105     }
106
107     /// @dev Allows the owner to set the address of an EventFactory contract.
108     /// @param _contractAddress The address of the EventFactory contract.
109     /*@CTK set_event_factory_address_success
110         @pre owner == msg.sender
111         @pre _contractAddress != address(0)
112         @post __post.whitelistedContracts[_contractAddress] == true
113         @post __post.eventFactoryVersionToAddress[currentEventFactoryIndex] ==
            _contractAddress
114         @post __post.eventFactoryAddressToVersion[_contractAddress] ==
            currentEventFactoryIndex
115         @post __post.currentEventFactoryIndex == currentEventFactoryIndex + 1
116     */
117     function setEventFactoryAddress(address _contractAddress)
118     public
119     onlyOwner()
120     validAddress(_contractAddress)
121     {
122         uint16 index = currentEventFactoryIndex;
123         eventFactoryVersionToAddress[index] = _contractAddress;
124         eventFactoryAddressToVersion[_contractAddress] = index;
125         currentEventFactoryIndex++;
126
127         whitelistedContracts[_contractAddress] = true;
128
129         EventFactoryAddressAdded(index, _contractAddress);
130         ContractWhitelisted(_contractAddress);
131     }
132
133     /// @dev Allows the owner to set the version of the next EventFactory. In case
134     /// AddressManager ever gets
135     /// upgraded, we need to be able to continue where the last version was.
136     /// @param _newIndex The index of where the next EventFactory version should start
137     .
138     function setCurrentEventFactoryIndex(uint16 _newIndex)
139     public
140     onlyOwner()

```

```

139 {
140     currentEventFactoryIndex = _newIndex;
141 }
142
143 /// @dev Allows the owner to set the address of an OracleFactory contract.
144 /// @param _contractAddress The address of the OracleFactory contract.
145 /*@CTK set_oracle_factory_address_success
146     @pre owner == msg.sender
147     @pre _contractAddress != address(0)
148     @post __post.whitelistedContracts[_contractAddress] == true
149     @post __post.oracleFactoryVersionToAddress[currentOracleFactoryIndex] ==
        _contractAddress
150     @post __post.oracleFactoryAddressToVersion[_contractAddress] ==
        currentOracleFactoryIndex
151     @post __post.currentOracleFactoryIndex == currentOracleFactoryIndex + 1
152 */
153 function setOracleFactoryAddress(address _contractAddress)
154     public
155     onlyOwner()
156     validAddress(_contractAddress)
157 {
158     uint16 index = currentOracleFactoryIndex;
159     oracleFactoryVersionToAddress[index] = _contractAddress;
160     oracleFactoryAddressToVersion[_contractAddress] = index;
161     currentOracleFactoryIndex++;
162
163     whitelistedContracts[_contractAddress] = true;
164
165     OracleFactoryAddressAdded(index, _contractAddress);
166     ContractWhitelisted(_contractAddress);
167 }
168
169 /// @dev Allows the owner to set the version of the next OracleFactory. In case
    AddressManager ever gets
170 /// upgraded, we need to be able to continue where the last version was.
171 /// @param _newIndex The index of where the next OracleFactory version should
    start.
172 /*@CTK set_current_oracle_factory_index_success
173     @pre owner == msg.sender
174     @post __post.currentOracleFactoryIndex == _newIndex
175 */
176 function setCurrentOracleFactoryIndex(uint16 _newIndex)
177     public
178     onlyOwner()
179 {
180     currentOracleFactoryIndex = _newIndex;
181 }
182
183 /*
184 * @dev Sets the eventEscrowAmount that is needed to create an Event.
185 * @param _newEscrowAmount The new escrow amount needed to create an Event.
186 */
187 /*@CTK set_event_escrow_amount
188     @pre owner == msg.sender
189     @post __post.eventEscrowAmount == _newEscrowAmount
190 */
191 function setEventEscrowAmount(uint256 _newEscrowAmount)
192     public

```

```

193     onlyOwner()
194     {
195         eventEscrowAmount = _newEscrowAmount;
196     }
197
198     /*
199     * @dev Sets the arbitrationLength that DecentralizedOracles will use.
200     * @param _newLength The new length in seconds (unix time) of an arbitration period
201     */
202     /*@CTK set_arbitration_length
203     @tag assume_completion
204     @pre owner == msg.sender
205     @post __post.arbitrationLength == _newLength
206     */
207     function setArbitrationLength(uint256 _newLength)
208     public
209     onlyOwner()
210     {
211         require(_newLength > 0);
212
213         arbitrationLength = _newLength;
214     }
215
216     /*
217     * @dev Sets the startingOracleThreshold that CentralizedOracles will use.
218     * @param _newThreshold The new consensusThreshold for CentralizedOracles.
219     */
220     /*@CTK set_starting_oracle_threshold
221     @pre owner == msg.sender
222     @post __post.startingOracleThreshold == _newThreshold
223     */
224     function setStartingOracleThreshold(uint256 _newThreshold)
225     public
226     onlyOwner()
227     {
228         startingOracleThreshold = _newThreshold;
229     }
230
231     /*
232     * @dev Sets the thresholdPercentIncrease that DecentralizedOracles will use.
233     * @param _newIncrement The new increment amount for DecentralizedOracles.
234     */
235     /*@CTK set_consensus_threshold_percent_increase
236     @pre owner == msg.sender
237     @post __post.thresholdPercentIncrease == _newPercentage
238     */
239     function setConsensusThresholdPercentIncrease(uint256 _newPercentage)
240     public
241     onlyOwner()
242     {
243         thresholdPercentIncrease = _newPercentage;
244     }
245
246     /// @notice Gets the latest index of a deployed EventFactory contract.
247     /// @return The index of the latest deployed EventFactory contract.
248     function getLastEventFactoryIndex()
249     public

```

```

250     view
251     returns (uint16 lastEventFactoryIndex)
252     {
253         if (currentEventFactoryIndex == 0) {
254             return 0;
255         } else {
256             return currentEventFactoryIndex - 1;
257         }
258     }
259
260     /// @notice Gets the latest index of a deployed OracleFactory contract.
261     /// @return The index of the latest deployed OracleFactory contract.
262     function getLastOracleFactoryIndex()
263     public
264     view
265     returns (uint16 lastOracleFactoryIndex)
266     {
267         if (currentOracleFactoryIndex == 0) {
268             return 0;
269         } else {
270             return currentOracleFactoryIndex - 1;
271         }
272     }
273 }

```

File libs/Ownable.sol

```

1  pragma solidity ^0.4.15;
2
3  /**
4   * @title Ownable contract
5   * @dev The Ownable contract has an owner address, and provides basic authorization
6       control functions.
7   */
8  contract Ownable {
9
10     // Modifiers
11     modifier onlyOwner() {
12         require(msg.sender == owner);
13         _;
14     }
15
16     modifier validAddress(address _address) {
17         require(_address != address(0));
18         _;
19     }
20
21     // Events
22     event OwnershipTransferred(address indexed _previousOwner, address indexed _newOwner
23         );
24
25     /// @dev The Ownable constructor sets the original 'owner' of the contract to the
26         sender account.
27     /*@CTK throw_on_invalid_address
28         @post _owner == address(0) -> __reverted == true
29     */
30     /*@CTK owner_set_on_success
31         @pre __reverted == false -> __post.owner == _owner

```

```

30  */
31  function Ownable(address _owner) public validAddress(_owner) {
32      owner = _owner;
33  }
34
35  /// @dev Allows the current owner to transfer control of the contract to a newOwner.
36  /// @param _newOwner The address to transfer ownership to.
37  /*@CTK transferOwnership
38      @post __reverted == false -> (msg.sender == owner -> __post.owner == _newOwner)
39      @post (owner != msg.sender) -> (__reverted == true)
40      @post (_newOwner == address(0)) -> (__reverted == true)
41  */
42  function transferOwnership(address _newOwner) public onlyOwner validAddress(
    _newOwner) {
43      OwnershipTransferred(owner, _newOwner);
44      owner = _newOwner;
45  }
46  }

```

File libs/SafeMath.sol

```

1  pragma solidity ^0.4.11;
2
3  library SafeMath {
4      /*@CTK SafeMath_add
5          @tag spec
6          @post __reverted == __has_assertion_failure
7          @post __has_assertion_failure == __has_overflow
8          @post __reverted == false -> __return == x + y
9          @post msg == msg__post
10         @post ((x + y < x) || (x + y < y)) == __has_assertion_failure
11         @post __addr_map == __addr_map__post
12     */
13     function add(uint256 x, uint256 y) internal pure returns(uint256) {
14         uint256 z = x + y;
15         assert((z >= x) && (z >= y));
16         return z;
17     }
18
19     /*@CTK SafeMath_sub
20         @tag spec
21         @post __reverted == __has_assertion_failure
22         @post __has_overflow == true -> __has_assertion_failure == true
23         @post __reverted == false -> __return == x - y
24         @post msg == msg__post
25         @post (x < y) == __has_assertion_failure
26         @post __addr_map == __addr_map__post
27     */
28     function sub(uint256 x, uint256 y) internal pure returns(uint256) {
29         assert(x >= y);
30         uint256 z = x - y;
31         return z;
32     }
33
34     /*@CTK SafeMath_mul
35         @tag spec
36         @post __reverted == __has_assertion_failure
37         @post __has_assertion_failure == __has_overflow
38         @post __reverted == false -> __return == x * y

```

```

39     @post msg == msg__post
40     @post (x > 0 && (x * y / x != y)) == __has_assertion_failure
41     @post __addr_map == __addr_map__post
42     */
43     function mul(uint256 x, uint256 y) internal pure returns(uint256) {
44         uint256 z = x * y;
45         assert((x == 0) || (z / x == y));
46         return z;
47     }
48
49     /*@CTK SafeMath_div
50     @tag spec
51     @post __reverted == __has_assertion_failure
52     @post y == 0 -> __has_assertion_failure == true
53     @post __has_overflow == true -> __has_assertion_failure == true
54     @post __reverted == false -> __return == x / y
55     @post msg == msg__post
56     @post (y == 0) == __has_assertion_failure
57     @post __addr_map == __addr_map__post
58     */
59     function div(uint256 x, uint256 y) internal pure returns(uint256) {
60         assert(y != 0);
61         uint256 z = x / y;
62         assert(x == y * z + x % y);
63         return z;
64     }
65 }

```

File tokens/BodhiToken.sol

```

1  pragma solidity ^0.4.17;
2
3  import './StandardToken.sol';
4  import './libs/Ownable.sol';
5
6  contract BodhiToken is StandardToken, Ownable {
7      // Token configurations
8      string public constant name = "Bodhi Token";
9      string public constant symbol = "BOT";
10     uint256 public constant decimals = 8;
11
12     uint256 public constant tokenTotalSupply = 100 * (10**6) * (10**decimals); // 100
        million BOT ever created
13
14     // Events
15     event Mint(uint256 supply, address indexed to, uint256 amount);
16
17     /// @notice Creates new BodhiToken contract
18     function BodhiToken() Ownable(msg.sender) public {
19     }
20
21     /// @notice Allows the owner to mint new tokens
22     /// @param _to Address to mint the tokens to
23     /// @param _amount Amount of tokens that will be minted
24     /// @return Boolean to signify successful minting
25
26     /*@CTK mintByOwner_check
27     @post msg.sender != owner -> __reverted == true
28     */

```



```

29  /*@CTK mintByOwner
30      @tag assume_completion
31      @post __post.balances[_to] == balances[_to] + _amount
32      @post __post.totalSupply == totalSupply + _amount
33  */
34  function mintByOwner(address _to, uint256 _amount) public onlyOwner returns (bool) {
35      return mint(_to, _amount);
36  }
37
38  /// @dev Mint new tokens
39  /// @param _to Address to mint the tokens to
40  /// @param _amount Amount of tokens that will be minted
41  /// @return Boolean to signify successful minting
42  /*@CTK mintCheck
43      @tag assume_completion
44      @post __has_overflow == false
45      @post __post.balances[_to] == balances[_to] + _amount
46      @post __post.totalSupply == totalSupply + _amount
47      @post __return == true
48  */
49  function mint(address _to, uint256 _amount) internal returns (bool) {
50      uint256 checkedSupply = totalSupply.add(_amount);
51      require(checkedSupply <= tokenTotalSupply);
52
53      totalSupply += _amount;
54      balances[_to] = balances[_to].add(_amount);
55
56      Mint(totalSupply, _to, _amount);
57
58      return true;
59  }
60 }

```

File tokens/StandardToken.sol

```

1  pragma solidity ^0.4.11;
2
3  import './BasicToken.sol';
4  import './ERC20.sol';
5
6  /**
7   * @title Standard ERC20 token
8   *
9   * @dev Implementation of the basic standard token.
10  * @dev https://github.com/ethereum/EIPs/issues/20
11  * @dev Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master/smart\_contract/FirstBloodToken.sol
12  */
13  contract StandardToken is ERC20, BasicToken {
14      mapping (address => mapping (address => uint256)) allowed;
15
16      /**
17       * @dev Transfer tokens from one address to another
18       * @param _from address The address which you want to send tokens from
19       * @param _to address The address which you want to transfer to
20       * @param _value uint256 the amount of tokens to be transferred
21       */
22      /*@CTK transferFrom
23          @tag assume_completion

```

```

24     @pre _from != _to
25     @post __return == true
26     @post __post.balances[_to] == balances[_to] + _value
27     @post __post.balances[_from] == balances[_from] - _value
28     @post __has_overflow == false
29 */
30 function transferFrom(address _from, address _to,
31     uint256 _value) public returns (bool) {
32     require(_to != address(0));
33     var _allowance = allowed[_from][msg.sender];
34
35     // Check is not needed because sub(_allowance, _value) will already throw if this
36     // condition is not met
37     // require (_value <= _allowance);
38
39     balances[_from] = balances[_from].sub(_value);
40     balances[_to] = balances[_to].add(_value);
41     allowed[_from][msg.sender] = _allowance.sub(_value);
42     Transfer(_from, _to, _value);
43     return true;
44 }
45
46 /**
47  * @dev Approve the passed address to spend the specified amount of tokens on behalf
48  *       of msg.sender.
49  * @param _spender The address which will spend the funds.
50  * @param _value The amount of tokens to be spent.
51  */
52 /*@CTK approve_success
53  @post _value == 0 -> __reverted == false
54  @post allowed[msg.sender][_spender] == 0 -> __reverted == false
55 */
56 /*@CTK approve
57  @tag assume_completion
58  @post __post.allowed[msg.sender][_spender] == _value
59 */
60 function approve(address _spender, uint256 _value) public returns (bool) {
61     // To change the approve amount you first have to reduce the addresses'
62     // allowance to zero by calling 'approve(_spender, 0)' if it is not
63     // already 0 to mitigate the race condition described here:
64     // https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
65     require((_value == 0) || (allowed[msg.sender][_spender] == 0));
66
67     allowed[msg.sender][_spender] = _value;
68     Approval(msg.sender, _spender, _value);
69     return true;
70 }
71
72 /**
73  * @dev Function to check the amount of tokens that an owner allowed to a spender.
74  * @param _owner address The address which owns the funds.
75  * @param _spender address The address which will spend the funds.
76  * @return A uint256 specifying the amount of tokens still available for the spender
77  */
78 /*@CTK get_allowance
79  @post __reverted == false
80  @post remaining == allowed[_owner][_spender]

```

```

79     @post this == __post
80     */
81     function allowance(address _owner, address _spender) public view returns (uint256
      remaining) {
82         return allowed[_owner][_spender];
83     }
84 }

```

File tokens/BasicToken.sol

```

1  pragma solidity ^0.4.17;
2
3  import './ERC20Basic.sol';
4  import '../libs/SafeMath.sol';
5
6  /**
7   * @title Basic token
8   * @dev Basic version of StandardToken, with no allowances.
9   */
10 contract BasicToken is ERC20Basic {
11     using SafeMath for uint256;
12
13     mapping(address => uint256) balances;
14
15     /**
16      * @dev transfer token for a specified address
17      * @param _to The address to transfer to.
18      * @param _value The amount to be transferred.
19      */
20     /*CTK transfer_success
21         @pre _to != address(0)
22         @pre balances[msg.sender] >= _value
23         @pre __reverted == false
24         @post __reverted == false
25         @post __return == true
26     */
27     /*CTK transfer_conditions
28         @tag assume_completion
29         @pre _to != msg.sender
30         @post __post.balances[_to] == balances[_to] + _value
31         @post __post.balances[msg.sender] == balances[msg.sender] - _value
32     */
33     /*CTK transfer_same_address
34         @tag assume_completion
35         @tag no_overflow
36         @pre _to == msg.sender
37         @post this == __post
38     */
39     function transfer(address _to, uint256 _value) public returns (bool) {
40         require(_to != address(0));
41
42         // SafeMath.sub will throw if there is not enough balance.
43         balances[msg.sender] = balances[msg.sender].sub(_value);
44         balances[_to] = balances[_to].add(_value);
45         Transfer(msg.sender, _to, _value);
46         return true;
47     }
48
49     /**

```

```

50  * @dev Gets the balance of the specified address.
51  * @param _owner The address to query the the balance of.
52  * @return An uint256 representing the amount owned by the passed address.
53  */
54
55  /*@CTK balanceOf
56   @post __reverted == false
57   @post balance == balances[_owner]
58  */
59  function balanceOf(address _owner) public view returns (uint256 balance) {
60      return balances[_owner];
61  }
62  }

```

File tokens/CrowdsaleBodhiToken.sol

```

1  pragma solidity ^0.4.18;
2
3  import './BodhiToken.sol';
4
5  contract CrowdsaleBodhiToken is BodhiToken {
6      uint256 public constant nativeDecimals = 18;
7
8      /// @notice 60 million BOT tokens for sale
9      uint256 public constant saleAmount = 60 * (10**6) * (10**decimals);
10
11     // Crowdsale parameters
12     uint256 public fundingStartBlock;
13     uint256 public fundingEndBlock;
14     uint256 public initialExchangeRate;
15
16     // Events
17     event TokenPurchase(address indexed purchaser, address indexed beneficiary,
18         uint256 value, uint256 amount);
19
20     /// @notice Creates new CrowdsaleBodhiToken contract
21     /// @param _fundingStartBlock The starting block of crowdsale
22     /// @param _fundingEndBlock The ending block of crowdsale
23     /// @param _initialExchangeRate The exchange rate of Ether to BOT
24     /// @param _presaleAmount The amount of BOT that will be available for presale
25     /*@CTK CrowdsaleBodhiToken
26      @pre __reverted == false
27      @pre balances[owner] == 0
28      @pre totalSupply == 0
29      @pre decimals == 8
30      @post __post.fundingStartBlock == _fundingStartBlock
31      @post __post.fundingEndBlock == _fundingEndBlock
32      @post __post.initialExchangeRate == _initialExchangeRate
33      @post __post.balances[owner] == __post.totalSupply
34    */
35    function CrowdsaleBodhiToken(
36        uint256 _fundingStartBlock,
37        uint256 _fundingEndBlock,
38        uint256 _initialExchangeRate,
39        uint256 _presaleAmount)
40    public
41    {
42        require(_fundingStartBlock >= block.number);
43        require(_fundingEndBlock >= _fundingStartBlock);

```

```

43     require(_initialExchangeRate > 0);
44
45     // Converted to lowest denomination of BOT
46     uint256 presaleAmountTokens = _presaleAmount * (10**decimals);
47     require(presaleAmountTokens <= saleAmount);
48
49     assert(nativeDecimals >= decimals);
50
51     fundingStartBlock = _fundingStartBlock;
52     fundingEndBlock = _fundingEndBlock;
53     initialExchangeRate = _initialExchangeRate;
54
55     // Mint the presale tokens, distribute to a receiver
56     // Increase the totalSupply accordingly
57     mintByOwner(owner, presaleAmountTokens);
58 }
59
60 /// @notice Fallback function to purchase tokens
61 function() external payable {
62     buyTokens(msg.sender);
63 }
64
65 /// @notice Allows buying tokens from different address than msg.sender
66 /// @param _beneficiary Address that will contain the purchased tokens
67 function buyTokens(address _beneficiary) public payable {
68     require(_beneficiary != address(0));
69     require(block.number >= fundingStartBlock);
70     require(block.number <= fundingEndBlock);
71     require(msg.value > 0);
72
73     uint256 tokenAmount = getTokenExchangeAmount(msg.value, initialExchangeRate,
74         nativeDecimals, decimals);
75     uint256 checkedSupply = totalSupply.add(tokenAmount);
76
77     // Ensure new token increment does not exceed the sale amount
78     assert(checkedSupply <= saleAmount);
79
80     mintByPurchaser(_beneficiary, tokenAmount);
81     TokenPurchase(msg.sender, _beneficiary, msg.value, tokenAmount);
82     owner.transfer(msg.value);
83 }
84
85 /// @notice Shows the amount of BOT the user will receive for amount of exchanged
86 /// wei
87 /// @param _weiAmount Exchanged wei amount to convert
88 /// @param _exchangeRate Number of BOT per exchange token
89 /// @param _nativeDecimals Number of decimals of the token being exchange for BOT
90 /// @param _decimals Number of decimals of BOT token
91 /// @return The amount of BOT that will be received
92 function getTokenExchangeAmount(
93     uint256 _weiAmount,
94     uint256 _exchangeRate,
95     uint256 _nativeDecimals,
96     uint256 _decimals)
97     public
98     pure
99     returns(uint256)
100 {

```

```
99     require(_weiAmount > 0);
100
101     uint256 differenceFactor = (10**_nativeDecimals) / (10**_decimals);
102     return _weiAmount.mul(_exchangeRate).div(differenceFactor);
103 }
104
105 /// @dev Function to allow crowdsale participants to mint tokens when purchasing
106 /// @param _to Address to mint the tokens to
107 /// @param _amount Amount of tokens that will be minted
108 /// @return Boolean to signify successful minting
109 /*@CTK mintByPurchaser
110   @pre __reverted == false
111   @post __post.balances[_to] == balances[_to] + _amount
112   @post __post.totalSupply == totalSupply + _amount
113   */
114 function mintByPurchaser(address _to, uint256 _amount) private returns (bool) {
115     return mint(_to, _amount);
116 }
117 }
```

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
Post environment	52	}
	53	balance: 0x0
	54	}
	55	}
	56	
	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

Static Analysis Request

INSECURE_COMPILER_VERSION

Line 1 in File BaseContract.sol

```
1 pragma solidity ^0.4.18;
```

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File Migrations.sol

```
1 pragma solidity ^0.4.15;
```

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File StandardTokenMock.sol

```
1 pragma solidity ^0.4.18;
```

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File BasicTokenMock.sol

```
1 pragma solidity ^0.4.18;
```

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File CentralizedOracle.sol


```
1 pragma solidity ^0.4.18;
```

 Only these compiler versions are safe to compile your code: 0.4.25

TIMESTAMP_DEPENDENCY

Line 95 in File CentralizedOracle.sol


```
95 require(block.timestamp >= bettingStartTime);
```

 "block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 96 in File CentralizedOracle.sol

```
96 require(block.timestamp < bettingEndTime);
```

 "block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 124 in File CentralizedOracle.sol

```
124     require(block.timestamp >= resultSettingStartTime);
```

! "block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 125 in File CentralizedOracle.sol

```
125     if (block.timestamp < resultSettingEndTime) {
```

! "block.timestamp" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File OracleFactory.sol

```
1 pragma solidity ^0.4.18;
```

i Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File DecentralizedOracle.sol

```
1 pragma solidity ^0.4.18;
```

i Only these compiler versions are safe to compile your code: 0.4.25

TIMESTAMP_DEPENDENCY

Line 47 in File DecentralizedOracle.sol

```
47     require(_arbitrationEndTime > block.timestamp);
```

! "block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 69 in File DecentralizedOracle.sol

```
69     require(block.timestamp < arbitrationEndTime);
```

! "block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 100 in File DecentralizedOracle.sol

```
100    require(block.timestamp >= arbitrationEndTime);
```

! "block.timestamp" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File AddressManager.sol

```
1 pragma solidity ^0.4.18;
```

i Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
1 pragma solidity ^0.4.15;
```

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
1 pragma solidity ^0.4.11;
```

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File BodhiToken.sol


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

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File StandardToken.sol

```
1 pragma solidity ^0.4.11;
```

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File BasicToken.sol

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

 Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File CrowdsaleBodhiToken.sol


```
1 pragma solidity ^0.4.18;
```

 Only these compiler versions are safe to compile your code: 0.4.25

Formal Verification Request 1

get_bet_balances

 03, Mar 2019

 16.69 ms

Line 29-32 in File BaseContract.sol

```
29  /*@CTK get_bet_balances
30  @tag spec
31  @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j]
32  */
```

Line 33-53 in File BaseContract.sol


```
33  function getBetBalances()
34  public
35  view
36  returns (uint256[11])
37  {
38  uint256[11] memory betBalances;
39  /*@CTK set_bet_balances
40  @var uint8 i
41  @var BaseContract this
42  @var uint256[11] betBalances
43  @inv forall j: uint. (j >= 0 /\ j < i) -> betBalances[j] == this.balances[j].
44  bets[msg.sender]
45  @inv i <= this.numOfResults
46  @inv this == this__pre
47  @post i >= numOfResults
48  @post !__should_return
49  */
50  for (uint8 i = 0; i < numOfResults; i++) {
51  betBalances[i] = balances[i].bets[msg.sender];
52  }
53  return betBalances;
}
```

 The code meets the specification

Formal Verification Request 2

get_total_bets

 03, Mar 2019

 16.35 ms

Line 59-62 in File BaseContract.sol

```
59  /*@CTK get_total_bets
60  @tag spec
61  @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j]
62  totalBets
```

Line 63-83 in File BaseContract.sol

```

63     function getTotalBets()
64     public
65     view
66     returns (uint256[11])
67     {
68         uint256[11] memory totalBets;
69         /*@CTK set_total_bets
70          @var uint8 i
71          @var BaseContract this
72          @var uint256[11] totalBets
73          @inv forall j: uint. (j >= 0 /\ j < i) -> totalBets[j] == this.balances[j].
              totalBets
74          @inv i <= this.numOfResults
75          @inv this == this__pre
76          @post i >= numOfResults
77          @post !__should_return
78          */
79         for (uint8 i = 0; i < numOfResults; i++) {
80             totalBets[i] = balances[i].totalBets;
81         }
82         return totalBets;
83     }

```

✓ The code meets the specification

Formal Verification Request 3

get_vote_balances



03, Mar 2019



17.29 ms

Line 89-92 in File BaseContract.sol

```

89     /*@CTK get_vote_balances
90     @tag spec
91     @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j]
              ].votes[msg.sender]
92     */

```

Line 93-113 in File BaseContract.sol

```

93     function getVoteBalances()
94     public
95     view
96     returns (uint256[11])
97     {
98         uint256[11] memory voteBalances;
99         /*@CTK set_vote_balances
100          @var uint8 i
101          @var BaseContract this
102          @var uint256[11] voteBalances
103          @inv forall j: uint. (j >= 0 /\ j < i) -> voteBalances[j] == this.balances[j]
              ].votes[msg.sender]
104          @inv i <= this.numOfResults

```

```

105     @inv this == this__pre
106     @post i >= numOfResults
107     @post !__should_return
108     */
109     for (uint8 i = 0; i < numOfResults; i++) {
110         voteBalances[i] = balances[i].votes[msg.sender];
111     }
112     return voteBalances;
113 }

```

✓ The code meets the specification

Formal Verification Request 4

get_total_votes

📅 03, Mar 2019

🕒 15.74 ms

Line 119-122 in File BaseContract.sol

```

119     /*@CTK get_total_votes
120     @tag spec
121     @post forall j: uint. (j >= 0 /\ j < numOfResults) -> __return[j] == balances[j
122         ].totalVotes
123     */

```

Line 123-143 in File BaseContract.sol

```

123     function getTotalVotes()
124     public
125     view
126     returns (uint256[11])
127     {
128         uint256[11] memory totalVotes;
129         /*@CTK set_total_votes
130         @var uint8 i
131         @var BaseContract this
132         @var uint256[11] totalVotes
133         @inv forall j: uint. (j >= 0 /\ j < i) -> totalVotes[j] == this.balances[j].
134             totalVotes
135         @inv i <= this.numOfResults
136         @inv this == this__pre
137         @post i >= numOfResults
138         @post !__should_return
139         */
140         for (uint8 i = 0; i < numOfResults; i++) {
141             totalVotes[i] = balances[i].totalVotes;
142         }
143         return totalVotes;


```

✓ The code meets the specification

Formal Verification Request 5

set_bet_balances__Generated

 03, Mar 2019

 115.68 ms

(Loop) Line 39-48 in File BaseContract.sol

39 No Snippet Available

(Loop) Line 39-51 in File BaseContract.sol


39 No Snippet Available

 The code meets the specification

Formal Verification Request 6

set_total_bets__Generated

 03, Mar 2019

 78.59 ms

(Loop) Line 69-78 in File BaseContract.sol

69 No Snippet Available

(Loop) Line 69-81 in File BaseContract.sol


69 No Snippet Available

 The code meets the specification

Formal Verification Request 7

set_vote_balances__Generated

 03, Mar 2019

 85.26 ms

(Loop) Line 99-108 in File BaseContract.sol

99 No Snippet Available

(Loop) Line 99-111 in File BaseContract.sol

99 No Snippet Available

 The code meets the specification

Formal Verification Request 8

set_total_votes__Generated

📅 03, Mar 2019

🕒 96.54 ms

(Loop) Line 129-138 in File BaseContract.sol

129 No Snippet Available

(Loop) Line 129-141 in File BaseContract.sol

129 No Snippet Available

✅ The code meets the specification

Formal Verification Request 9

init_migrations

📅 03, Mar 2019

🕒 6.55 ms

Line 11-13 in File Migrations.sol

```
11  /*@CTK init_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 10

set_complete

📅 03, Mar 2019

🕒 8.56 ms

Line 18-21 in File Migrations.sol

```
18  /*@CTK set_complete
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

Formal Verification Request 11

Method will not encounter an assertion failure.

 03, Mar 2019

 28.69 ms

Line 26 in File Migrations.sol

```
26 // @CTK NO_ASF
```

Line 27-30 in File Migrations.sol


```
27 function upgrade(address new_address) public restricted {  
28     Migrations upgraded = Migrations(new_address);  
29     upgraded.setCompleted(last_completed_migration);  
30 }
```

✓ The code meets the specification

Formal Verification Request 12

init_mock_standard_token

 03, Mar 2019

 9.36 ms

Line 6-9 in File StandardTokenMock.sol

```
6 /* @CTK init_mock_standard_token  
7     @post __post.balances[_initialAccount] == _initialBalance  
8     @post __post.totalSupply == _initialBalance  
9 */
```

Line 10-14 in File StandardTokenMock.sol


```
10 function StandardTokenMock(address _initialAccount,  
11     uint256 _initialBalance) public {  
12     balances[_initialAccount] = _initialBalance;  
13     totalSupply = _initialBalance;  
14 }
```

✓ The code meets the specification

Formal Verification Request 13

init_mock_basic_token

 03, Mar 2019

 9.77 ms

Line 6-9 in File BasicTokenMock.sol


```

6  /*@CTK init_mock_basic_token
7      @post __post.balances[_initialAccount] == _initialBalance
8      @post __post.totalSupply == _initialBalance
9  */

```

Line 10-13 in File BasicTokenMock.sol

```

10  function BasicTokenMock(address _initialAccount, uint256 _initialBalance) public {
11      balances[_initialAccount] = _initialBalance;
12      totalSupply = _initialBalance;
13  }

```

✓ The code meets the specification

Formal Verification Request 14

CentralizedOracle constructor

📅 03, Mar 2019

⌚ 437.33 ms

Line 25-35 in File CentralizedOracle.sol

```

25  /*@CTK "CentralizedOracle constructor"
26      @tag assume_completion
27      @post __post.eventAddress == _eventAddress
28      @post __post.numOfResults == _numOfResults
29      @post __post.oracle == _oracle
30      @post __post.bettingStartTime == _bettingStartTime
31      @post __post.bettingEndTime == _bettingEndTime
32      @post __post.resultSettingStartTime == _resultSettingStartTime
33      @post __post.resultSettingEndTime == _resultSettingEndTime
34      @post __post.consensusThreshold == _consensusThreshold
35  */

```

Line 40-71 in File CentralizedOracle.sol

```

40  function CentralizedOracle(
41      uint16 _version,
42      address _owner,
43      address _eventAddress,
44      uint8 _numOfResults,
45      address _oracle,
46      uint256 _bettingStartTime,
47      uint256 _bettingEndTime,
48      uint256 _resultSettingStartTime,
49      uint256 _resultSettingEndTime,
50      uint256 _consensusThreshold)
51      Ownable(_owner)
52      public
53      validAddress(_oracle)
54      validAddress(_eventAddress)
55  {
56      require(_numOfResults > 0);
57      require(_bettingEndTime > _bettingStartTime);
58      require(_resultSettingStartTime >= _bettingEndTime);
59      require(_resultSettingEndTime > _resultSettingStartTime);
60      require(_consensusThreshold > 0);

```

```

61
62     version = _version;
63     eventAddress = _eventAddress;
64     numOfResults = _numOfResults;
65     oracle = _oracle;
66     bettingStartTime = _bettingStartTime;
67     bettingEndTime = _bettingEndTime;
68     resultSettingStartTime = _resultSettingStartTime;
69     resultSettingEndTime = _resultSettingEndTime;
70     consensusThreshold = _consensusThreshold;
71 }

```

✓ The code meets the specification

Formal Verification Request 15

contract failed with invalid input

📅 03, Mar 2019

🕒 260.29 ms

Line 36-39 in File CentralizedOracle.sol

```

36  /*@CTK "contract failed with invalid input"
37     @pre _numOfResults == 0 \/_consensusThreshold == 0
38     @post __reverted == true
39  */

```

Line 40-71 in File CentralizedOracle.sol

```

40  function CentralizedOracle(
41      uint16 _version,
42      address _owner,
43      address _eventAddress,
44      uint8 _numOfResults,
45      address _oracle,
46      uint256 _bettingStartTime,
47      uint256 _bettingEndTime,
48      uint256 _resultSettingStartTime,
49      uint256 _resultSettingEndTime,
50      uint256 _consensusThreshold)
51      Ownable(_owner)
52      public
53      validAddress(_oracle)
54      validAddress(_eventAddress)
55  {
56      require(_numOfResults > 0);
57      require(_bettingEndTime > _bettingStartTime);
58      require(_resultSettingStartTime >= _bettingEndTime);
59      require(_resultSettingEndTime > _resultSettingStartTime);
60      require(_consensusThreshold > 0);
61
62      version = _version;
63      eventAddress = _eventAddress;
64      numOfResults = _numOfResults;
65      oracle = _oracle;
66      bettingStartTime = _bettingStartTime;

```

```

67     bettingEndTime = _bettingEndTime;
68     resultSettingStartTime = _resultSettingStartTime;
69     resultSettingEndTime = _resultSettingEndTime;
70     consensusThreshold = _consensusThreshold;
71 }

```

✓ The code meets the specification

Formal Verification Request 16

OracleFactory constructor

📅 03, Mar 2019

🕒 14.97 ms

Line 38-40 in File OracleFactory.sol

```

38  /*@CTK "OracleFactory constructor"
39     @post __reverted == false -> (__post.addressManager == _addressManager)
40  */

```

Line 41-46 in File OracleFactory.sol

```

41  function OracleFactory(address _addressManager) public {
42      require(_addressManager != address(0));
43
44      addressManager = _addressManager;
45      // version = IAddressManager(addressManager).currentOracleFactoryIndex();
46  }

```

✓ The code meets the specification

Formal Verification Request 17

DecentralizedOracle constructor

📅 03, Mar 2019

🕒 220.57 ms

Line 19-29 in File DecentralizedOracle.sol

```

19  /*@CTK "DecentralizedOracle constructor"
20     @tag assume_completion
21     @pre _numOfResults > 0
22     @pre _consensusThreshold > 0
23     @post __post.version == _version
24     @post __post.eventAddress == _eventAddress
25     @post __post.numOfResults == _numOfResults
26     @post __post.lastResultIndex == _lastResultIndex
27     @post __post.arbitrationEndTime == _arbitrationEndTime
28     @post __post.consensusThreshold == _consensusThreshold
29  */

```

Line 34-56 in File DecentralizedOracle.sol


```
34 function DecentralizedOracle(  
35     uint16 _version,  
36     address _owner,  
37     address _eventAddress,  
38     uint8 _numOfResults,  
39     uint8 _lastResultIndex,  
40     uint256 _arbitrationEndTime,  
41     uint256 _consensusThreshold)  
42     Ownable(_owner)  
43     public  
44     validAddress(_eventAddress)  
45 {  
46     require(_numOfResults > 0);  
47     require(_arbitrationEndTime > block.timestamp);  
48     require(_consensusThreshold > 0);  
49  
50     version = _version;  
51     eventAddress = _eventAddress;  
52     numOfResults = _numOfResults;  
53     lastResultIndex = _lastResultIndex;  
54     arbitrationEndTime = _arbitrationEndTime;  
55     consensusThreshold = _consensusThreshold;  
56 }
```

✓ The code meets the specification

Formal Verification Request 18

whitelist_success

 03, Mar 2019

 34.91 ms

Line 72-76 in File AddressManager.sol

```
72 /*@CTK whitelist_success  
73     @pre whitelistedContracts[_contractAddress] == true  
74     @pre _contractAddress != address(0)  
75     @post __post.whitelistedContracts[_contractAddress] == true  
76 */
```

Line 77-85 in File AddressManager.sol


```
77 function addWhitelistContract(address _contractAddress)  
78     external  
79     isWhitelisted(msg.sender)  
80     validAddress(_contractAddress)  
81 {  
82     whitelistedContracts[_contractAddress] = true;  
83  
84     ContractWhitelisted(_contractAddress);  
85 }
```

✓ The code meets the specification

Formal Verification Request 19

set_bodhi_token_address_success

 03, Mar 2019

 40.05 ms

Line 89-94 in File AddressManager.sol

```

89  /*@CTK set_bodhi_token_address_success
90      @pre _tokenAddress != address(0)
91      @pre owner == msg.sender
92      @post __post.whitelistedContracts[_tokenAddress] == true
93      @post __post.bodhiTokenAddress == _tokenAddress
94  */

```

Line 95-105 in File AddressManager.sol

```

95  function setBodhiTokenAddress(address _tokenAddress)
96      public
97      onlyOwner()
98      validAddress(_tokenAddress)
99  {
100      bodhiTokenAddress = _tokenAddress;
101      whitelistedContracts[_tokenAddress] = true;
102
103      BodhiTokenAddressChanged(bodhiTokenAddress);
104      ContractWhitelisted(_tokenAddress);
105  }


```

 The code meets the specification

Formal Verification Request 20

set_event_factory_address_success

 03, Mar 2019

 70.89 ms

Line 109-116 in File AddressManager.sol

```

109  /*@CTK set_event_factory_address_success
110      @pre owner == msg.sender
111      @pre _contractAddress != address(0)
112      @post __post.whitelistedContracts[_contractAddress] == true
113      @post __post.eventFactoryVersionToAddress[currentEventFactoryIndex] ==
114          _contractAddress
114      @post __post.eventFactoryAddressToVersion[_contractAddress] ==
115          currentEventFactoryIndex
115      @post __post.currentEventFactoryIndex == currentEventFactoryIndex + 1
116  */

```

Line 117-131 in File AddressManager.sol

```

117  function setEventFactoryAddress(address _contractAddress)
118      public
119      onlyOwner()

```

```

120     validAddress(_contractAddress)
121     {
122         uint16 index = currentEventFactoryIndex;
123         eventFactoryVersionToAddress[index] = _contractAddress;
124         eventFactoryAddressToVersion[_contractAddress] = index;
125         currentEventFactoryIndex++;
126
127         whitelistedContracts[_contractAddress] = true;
128
129         EventFactoryAddressAdded(index, _contractAddress);
130         ContractWhitelisted(_contractAddress);
131     }

```

✓ The code meets the specification

Formal Verification Request 21

set_oracle_factory_address_success

📅 03, Mar 2019

🕒 68.32 ms

Line 145-152 in File AddressManager.sol

```

145     /*@CTK set_oracle_factory_address_success
146         @pre owner == msg.sender
147         @pre _contractAddress != address(0)
148         @post __post.whitelistedContracts[_contractAddress] == true
149         @post __post.oracleFactoryVersionToAddress[currentOracleFactoryIndex] ==
            _contractAddress
150         @post __post.oracleFactoryAddressToVersion[_contractAddress] ==
            currentOracleFactoryIndex
151         @post __post.currentOracleFactoryIndex == currentOracleFactoryIndex + 1
152     */

```

Line 153-167 in File AddressManager.sol

```

153     function setOracleFactoryAddress(address _contractAddress)
154     public
155     onlyOwner()
156     validAddress(_contractAddress)
157     {
158         uint16 index = currentOracleFactoryIndex;
159         oracleFactoryVersionToAddress[index] = _contractAddress;
160         oracleFactoryAddressToVersion[_contractAddress] = index;
161         currentOracleFactoryIndex++;
162
163         whitelistedContracts[_contractAddress] = true;
164
165         OracleFactoryAddressAdded(index, _contractAddress);
166         ContractWhitelisted(_contractAddress);
167     }


```

✓ The code meets the specification

Formal Verification Request 22

set_current_oracle_factory_index_success

 03, Mar 2019

 20.68 ms

Line 172-175 in File AddressManager.sol

```
172  /*@CTK set_current_oracle_factory_index_success
173      @pre owner == msg.sender
174      @post __post.currentOracleFactoryIndex == _newIndex
175  */
```

Line 176-181 in File AddressManager.sol


```
176  function setCurrentOracleFactoryIndex(uint16 _newIndex)
177      public
178      onlyOwner()
179  {
180      currentOracleFactoryIndex = _newIndex;
181  }
```

 The code meets the specification

Formal Verification Request 23

set_event_escrow_amount

 03, Mar 2019

 20.52 ms

Line 187-190 in File AddressManager.sol

```
187  /*@CTK set_event_escrow_amount
188      @pre owner == msg.sender
189      @post __post.eventEscrowAmount == _newEscrowAmount
190  */
```

Line 191-196 in File AddressManager.sol


```
191  function setEventEscrowAmount(uint256 _newEscrowAmount)
192      public
193      onlyOwner()
194  {
195      eventEscrowAmount = _newEscrowAmount;
196  }
```

 The code meets the specification

Formal Verification Request 24

set_arbitration_length

 03, Mar 2019

 29.33 ms

Line 202-206 in File AddressManager.sol

```
202  /*@CTK set_arbitration_length
203      @tag assume_completion
204      @pre owner == msg.sender
205      @post __post.arbitrationLength == _newLength
206  */
```

Line 207-214 in File AddressManager.sol

```
207  function setArbitrationLength(uint256 _newLength)
208      public
209      onlyOwner()
210  {
211      require(_newLength > 0);
212
213      arbitrationLength = _newLength;
214  }
```

✓ The code meets the specification

Formal Verification Request 25

set_starting_oracle_threshold

📅 03, Mar 2019

🕒 19.1 ms

Line 220-223 in File AddressManager.sol

```
220  /*@CTK set_starting_oracle_threshold
221      @pre owner == msg.sender
222      @post __post.startingOracleThreshold == _newThreshold
223  */
```

Line 224-229 in File AddressManager.sol

```
224  function setStartingOracleThreshold(uint256 _newThreshold)
225      public
226      onlyOwner()
227  {
228      startingOracleThreshold = _newThreshold;
229  }
```

✓ The code meets the specification

Formal Verification Request 26

set_consensus_threshold_percent_increase

📅 03, Mar 2019

🕒 19.95 ms

Line 235-238 in File AddressManager.sol


```
235  /*@CTK set_consensus_threshold_percent_increase
236      @pre owner == msg.sender
237      @post __post.thresholdPercentIncrease == _newPercentage
238  */
```

Line 239-244 in File AddressManager.sol


```
239  function setConsensusThresholdPercentIncrease(uint256 _newPercentage)
240      public
241      onlyOwner()
242  {
243      thresholdPercentIncrease = _newPercentage;
244  }
```

✓ The code meets the specification

Formal Verification Request 27

throw_on_invalid_address

 03, Mar 2019

 16.0 ms

Line 25-27 in File Ownable.sol

```
25  /*@CTK throw_on_invalid_address
26      @post _owner == address(0) -> __reverted == true
27  */
```

Line 31-33 in File Ownable.sol


```
31  function Ownable(address _owner) public validAddress(_owner) {
32      owner = _owner;
33  }
```

✓ The code meets the specification

Formal Verification Request 28

owner_set_on_success

 03, Mar 2019

 0.36 ms

Line 28-30 in File Ownable.sol

```
28  /*@CTK owner_set_on_success
29      @pre __reverted == false -> __post.owner == _owner
30  */
```

Line 31-33 in File Ownable.sol


```
31  function Ownable(address _owner) public validAddress(_owner) {
32      owner = _owner;
33  }
```

✓ The code meets the specification

Formal Verification Request 29

transferOwnership

 03, Mar 2019

 29.2 ms

Line 37-41 in File Ownable.sol

```

37  /*@CTK transferOwnership
38  @post __reverted == false -> (msg.sender == owner -> __post.owner == _newOwner)
39  @post (owner != msg.sender) -> (__reverted == true)
40  @post (_newOwner == address(0)) -> (__reverted == true)
41  */

```

Line 42-45 in File Ownable.sol

```

42  function transferOwnership(address _newOwner) public onlyOwner validAddress(
    _newOwner) {
43      OwnershipTransferred(owner, _newOwner);
44      owner = _newOwner;
45  }


```

 The code meets the specification

Formal Verification Request 30

SafeMath_add

 03, Mar 2019

 19.47 ms

Line 4-12 in File SafeMath.sol

```

4  /*@CTK SafeMath_add
5  @tag spec
6  @post __reverted == __has_assertion_failure
7  @post __has_assertion_failure == __has_overflow
8  @post __reverted == false -> __return == x + y
9  @post msg == msg__post
10 @post ((x + y < x) || (x + y < y)) == __has_assertion_failure
11 @post __addr_map == __addr_map__post
12 */

```

Line 13-17 in File SafeMath.sol

```

13 function add(uint256 x, uint256 y) internal pure returns(uint256) {
14     uint256 z = x + y;
15     assert((z >= x) && (z >= y));
16     return z;
17 }


```

 The code meets the specification

Formal Verification Request 31

SafeMath_sub

 03, Mar 2019

 15.21 ms

Line 19-27 in File SafeMath.sol

```
19  /*@CTK SafeMath_sub
20     @tag spec
21     @post __reverted == __has_assertion_failure
22     @post __has_overflow == true -> __has_assertion_failure == true
23     @post __reverted == false -> __return == x - y
24     @post msg == msg__post
25     @post (x < y) == __has_assertion_failure
26     @post __addr_map == __addr_map__post
27  */
```

Line 28-32 in File SafeMath.sol


```
28  function sub(uint256 x, uint256 y) internal pure returns(uint256) {
29      assert(x >= y);
30      uint256 z = x - y;
31      return z;
32  }
```

✓ The code meets the specification

Formal Verification Request 32

SafeMath_mul

 03, Mar 2019

 114.11 ms

Line 34-42 in File SafeMath.sol

```
34  /*@CTK SafeMath_mul
35     @tag spec
36     @post __reverted == __has_assertion_failure
37     @post __has_assertion_failure == __has_overflow
38     @post __reverted == false -> __return == x * y
39     @post msg == msg__post
40     @post (x > 0 && (x * y / x != y)) == __has_assertion_failure
41     @post __addr_map == __addr_map__post
42  */
```

Line 43-47 in File SafeMath.sol


```
43  function mul(uint256 x, uint256 y) internal pure returns(uint256) {
44      uint256 z = x * y;
45      assert((x == 0) || (z / x == y));
46      return z;
47  }
```

✓ The code meets the specification

Formal Verification Request 33

SafeMath_div

 03, Mar 2019

 1095.07 ms

Line 49-58 in File SafeMath.sol

```
49  /*@CTK SafeMath_div
50     @tag spec
51     @post __reverted == __has_assertion_failure
52     @post y == 0 -> __has_assertion_failure == true
53     @post __has_overflow == true -> __has_assertion_failure == true
54     @post __reverted == false -> __return == x / y
55     @post msg == msg__post
56     @post (y == 0) == __has_assertion_failure
57     @post __addr_map == __addr_map__post
58  */
```

Line 59-64 in File SafeMath.sol


```
59  function div(uint256 x, uint256 y) internal pure returns(uint256) {
60      assert(y != 0);
61      uint256 z = x / y;
62      assert(x == y * z + x % y);
63      return z;
64  }
```

 The code meets the specification

Formal Verification Request 34

mintByOwner_check

 03, Mar 2019

 109.23 ms

Line 26-28 in File BodhiToken.sol

```
26  /*@CTK mintByOwner_check
27     @post msg.sender != owner -> __reverted == true
28  */
```

Line 34-36 in File BodhiToken.sol


```
34  function mintByOwner(address _to, uint256 _amount) public onlyOwner returns (bool) {
35      return mint(_to, _amount);
36  }
```

 The code meets the specification

Formal Verification Request 35

mintByOwner

 03, Mar 2019

 141.09 ms

Line 29-33 in File BodhiToken.sol

```
29  /*@CTK mintByOwner
30     @tag assume_completion
31     @post __post.balances[_to] == balances[_to] + _amount
32     @post __post.totalSupply == totalSupply + _amount
33  */
```

Line 34-36 in File BodhiToken.sol


```
34  function mintByOwner(address _to, uint256 _amount) public onlyOwner returns (bool) {
35      return mint(_to, _amount);
36  }
```

 The code meets the specification

Formal Verification Request 36

mintCheck

 03, Mar 2019

 93.81 ms

Line 42-48 in File BodhiToken.sol

```
42  /*@CTK mintCheck
43     @tag assume_completion
44     @post __has_overflow == false
45     @post __post.balances[_to] == balances[_to] + _amount
46     @post __post.totalSupply == totalSupply + _amount
47     @post __return == true
48  */
```

Line 49-59 in File BodhiToken.sol


```
49  function mint(address _to, uint256 _amount) internal returns (bool) {
50      uint256 checkedSupply = totalSupply.add(_amount);
51      require(checkedSupply <= tokenTotalSupply);
52
53      totalSupply += _amount;
54      balances[_to] = balances[_to].add(_amount);
55
56      Mint(totalSupply, _to, _amount);
57
58      return true;
59  }
```

 The code meets the specification

Formal Verification Request 37

transferFrom

 03, Mar 2019

 87.08 ms

Line 22-29 in File StandardToken.sol

```
22  /*@CTK transferFrom
23      @tag assume_completion
24      @pre _from != _to
25      @post __return == true
26      @post __post.balances[_to] == balances[_to] + _value
27      @post __post.balances[_from] == balances[_from] - _value
28      @post __has_overflow == false
29  */
```

Line 30-43 in File StandardToken.sol


```
30  function transferFrom(address _from, address _to,
31      uint256 _value) public returns (bool) {
32      require(_to != address(0));
33      var _allowance = allowed[_from][msg.sender];
34
35      // Check is not needed because sub(_allowance, _value) will already throw if this
36      // condition is not met
37      // require (_value <= _allowance);
38
39      balances[_from] = balances[_from].sub(_value);
40      balances[_to] = balances[_to].add(_value);
41      allowed[_from][msg.sender] = _allowance.sub(_value);
42      Transfer(_from, _to, _value);
43      return true;
44  }
```

 The code meets the specification

Formal Verification Request 38

approve_success

 03, Mar 2019

 19.41 ms

Line 50-53 in File StandardToken.sol

```
50  /*@CTK approve_success
51      @post _value == 0 -> __reverted == false
52      @post allowed[msg.sender][_spender] == 0 -> __reverted == false
53  */
```

Line 58-68 in File StandardToken.sol

```
58  function approve(address _spender, uint256 _value) public returns (bool) {
59      // To change the approve amount you first have to reduce the addresses'
60      // allowance to zero by calling 'approve(_spender, 0)' if it is not
```

```
61 // already 0 to mitigate the race condition described here:
62 // https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
63 require((_value == 0) || (allowed[msg.sender][_spender] == 0));
64
65 allowed[msg.sender][_spender] = _value;
66 Approval(msg.sender, _spender, _value);
67 return true;
68 }
```

✓ The code meets the specification

Formal Verification Request 39

approve

📅 03, Mar 2019

🕒 1.67 ms

Line 54-57 in File StandardToken.sol

```
54 /*@CTK approve
55 @tag assume_completion
56 @post __post.allowed[msg.sender][_spender] == _value
57 */
```

Line 58-68 in File StandardToken.sol

```
58 function approve(address _spender, uint256 _value) public returns (bool) {
59 // To change the approve amount you first have to reduce the addresses'
60 // allowance to zero by calling 'approve(_spender, 0)' if it is not
61 // already 0 to mitigate the race condition described here:
62 // https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
63 require((_value == 0) || (allowed[msg.sender][_spender] == 0));
64
65 allowed[msg.sender][_spender] = _value;
66 Approval(msg.sender, _spender, _value);
67 return true;
68 }
```

✓ The code meets the specification

Formal Verification Request 40

get_allowance

📅 03, Mar 2019

🕒 5.93 ms

Line 76-80 in File StandardToken.sol

```
76 /*@CTK get_allowance
77 @post __reverted == false
78 @post remaining == allowed[_owner][_spender]
79 @post this == __post
80 */
```

Line 81-83 in File StandardToken.sol


```
81  function allowance(address _owner, address _spender) public view returns (uint256
      remaining) {
82      return allowed[_owner][_spender];
83  }
```

✓ The code meets the specification

Formal Verification Request 41

transfer_success

 03, Mar 2019

 36.65 ms

Line 20-26 in File BasicToken.sol

```
20  /*@CTK transfer_success
21      @pre _to != address(0)
22      @pre balances[msg.sender] >= _value
23      @pre __reverted == false
24      @post __reverted == false
25      @post __return == true
26  */
```

Line 39-47 in File BasicToken.sol


```
39  function transfer(address _to, uint256 _value) public returns (bool) {
40      require(_to != address(0));
41
42      // SafeMath.sub will throw if there is not enough balance.
43      balances[msg.sender] = balances[msg.sender].sub(_value);
44      balances[_to] = balances[_to].add(_value);
45      Transfer(msg.sender, _to, _value);
46      return true;
47  }
```

✓ The code meets the specification

Formal Verification Request 42

transfer_conditions

 03, Mar 2019

 34.97 ms

Line 27-32 in File BasicToken.sol

```
27  /*@CTK transfer_conditions
28      @tag assume_completion
29      @pre _to != msg.sender
30      @post __post.balances[_to] == balances[_to] + _value
31      @post __post.balances[msg.sender] == balances[msg.sender] - _value
32  */
```


Line 39-47 in File BasicToken.sol

```
39  function transfer(address _to, uint256 _value) public returns (bool) {
40      require(_to != address(0));
41
42      // SafeMath.sub will throw if there is not enough balance.
43      balances[msg.sender] = balances[msg.sender].sub(_value);
44      balances[_to] = balances[_to].add(_value);
45      Transfer(msg.sender, _to, _value);
46      return true;
47  }
```

✓ The code meets the specification

Formal Verification Request 43

transfer_same_address

📅 03, Mar 2019

🕒 5.63 ms

Line 33-38 in File BasicToken.sol

```
33  /*@CTK transfer_same_address
34      @tag assume_completion
35      @tag no_overflow
36      @pre _to == msg.sender
37      @post this == __post
38  */
```

Line 39-47 in File BasicToken.sol

```
39  function transfer(address _to, uint256 _value) public returns (bool) {
40      require(_to != address(0));
41
42      // SafeMath.sub will throw if there is not enough balance.
43      balances[msg.sender] = balances[msg.sender].sub(_value);
44      balances[_to] = balances[_to].add(_value);
45      Transfer(msg.sender, _to, _value);
46      return true;
47  }
```

✓ The code meets the specification

Formal Verification Request 44

balanceOf

📅 03, Mar 2019

🕒 6.19 ms

Line 55-58 in File BasicToken.sol

```

55  /*@CTK balanceOf
56      @post __reverted == false
57      @post balance == balances[_owner]
58  */

```

Line 59-61 in File BasicToken.sol

```

59  function balanceOf(address _owner) public view returns (uint256 balance) {
60      return balances[_owner];
61  }


```

✓ The code meets the specification

Formal Verification Request 45

CrowdsaleBodhiToken

 03, Mar 2019

 384.8 ms

Line 24-33 in File CrowdsaleBodhiToken.sol

```

24  /*@CTK CrowdsaleBodhiToken
25      @pre __reverted == false
26      @pre balances[owner] == 0
27      @pre totalSupply == 0
28      @pre decimals == 8
29      @post __post.fundingStartBlock == _fundingStartBlock
30      @post __post.fundingEndBlock == _fundingEndBlock
31      @post __post.initialExchangeRate == _initialExchangeRate
32      @post __post.balances[owner] == __post.totalSupply
33  */

```

Line 34-58 in File CrowdsaleBodhiToken.sol

```

34  function CrowdsaleBodhiToken(
35      uint256 _fundingStartBlock,
36      uint256 _fundingEndBlock,
37      uint256 _initialExchangeRate,
38      uint256 _presaleAmount)
39      public
40  {
41      require(_fundingStartBlock >= block.number);
42      require(_fundingEndBlock >= _fundingStartBlock);
43      require(_initialExchangeRate > 0);
44
45      // Converted to lowest denomination of BOT
46      uint256 presaleAmountTokens = _presaleAmount * (10**decimals);
47      require(presaleAmountTokens <= saleAmount);
48
49      assert(nativeDecimals >= decimals);
50
51      fundingStartBlock = _fundingStartBlock;
52      fundingEndBlock = _fundingEndBlock;
53      initialExchangeRate = _initialExchangeRate;
54
55      // Mint the presale tokens, distribute to a receiver
56      // Increase the totalSupply accordingly

```

```
57     mintByOwner(owner, presaleAmountTokens);  
58 }
```

✓ The code meets the specification

Formal Verification Request 46

mintByPurchaser

📅 03, Mar 2019

🕒 196.85 ms

Line 109-113 in File CrowdsaleBodhiToken.sol

```
109  /*@CTK mintByPurchaser  
110     @pre __reverted == false  
111     @post __post.balances[_to] == balances[_to] + _amount  
112     @post __post.totalSupply == totalSupply + _amount  
113     */
```

Line 114-116 in File CrowdsaleBodhiToken.sol

```
114  function mintByPurchaser(address _to, uint256 _amount) private returns (bool) {  
115      return mint(_to, _amount);  
116  }
```

✓ The code meets the specification