

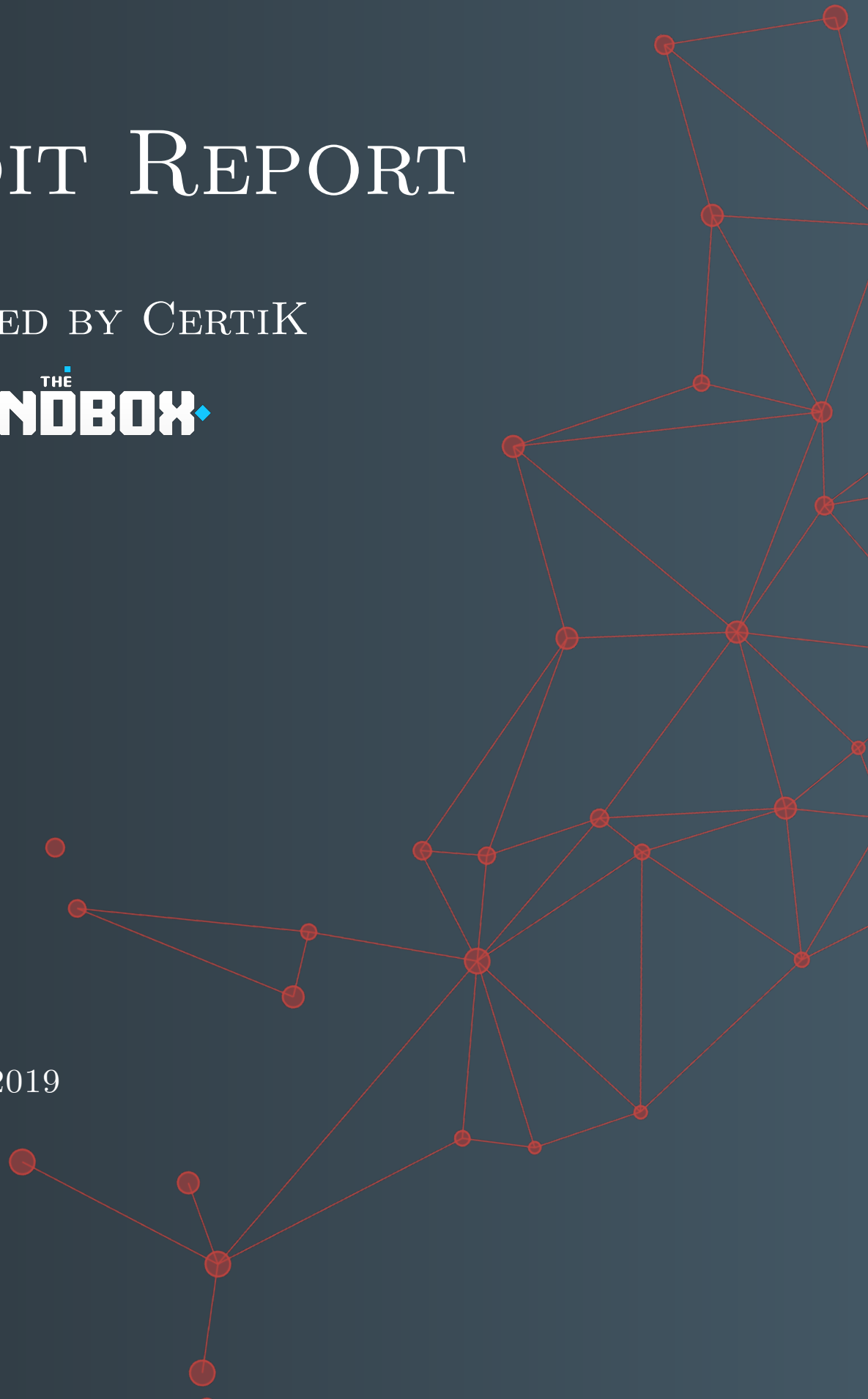


AUDIT REPORT

PRODUCED BY CERTIK

FOR **THE SANDBOX**

10TH DEC, 2019



CERTIK AUDIT REPORT FOR THE SANDBOX



Request Date: 2019-11-08
Revision Date: 2019-12-10
Platform Name: Ethereum



Contents

Disclaimer	1
About CertiK	2
Executive Summary	3
Vulnerability Classification	3
Testing Summary	4
Audit Score	4
Type of Issues	4
Vulnerability Details	5
Manual Review Notes	6
Static Analysis Results	9
Formal Verification Results	11
How to read	11
Source Code with CertiK Labels	267

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 The Sandbox(the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

About CertiK

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

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

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

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

Executive Summary

This report has been prepared for The Sandbox to discover issues and vulnerabilities in the source code of their LandBaseToken, Land and LandSaleWithETHAndDAI smart contract. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Vulnerability Classification

CertiK categorizes issues into three buckets based on overall risk levels:

Critical

Code implementation does not match specification, which could result in the loss of funds for contract owner or users.

Medium

Code implementation does not match the specification under certain conditions, which could affect the security standard by loss of access control.

Low

Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.

Testing Summary

PASS

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

Dec 10, 2019



Type of Issues

CertiK's smart label engine applied 100% formal verification coverage on the source code. Our team of engineers has scanned the source code using proprietary static analysis tools and code-review methodologies. The following technical issues were found:

Title	Description	Issues	SWC ID
Integer Overflow and Underflow Function Incorrectness	An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Buffer Overflow	An attacker can 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 miners to some degree.	0	SWC-116
Insecure Compiler Version	Using a 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	Using block attributes to generate random numbers is unreliable, as they can be influenced by miners to some degree.	0	SWC-120
“tx.origin” for Authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling untrusted contracts is very dangerous, so the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default, meaning a malicious user can 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.	0	SWC-111
Unused Variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

ERC721BaseToken:

- `burnFrom(from, id)`: When item `id`'s operator is set to owner, `burnFrom` function enables anyone to burn the item.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Source Code SHA-256 Checksum¹

- **AddressUtils.sol**
2a717cd56c8a3f562015bacb0ab7b6d93cb639d64221728520bf3f40217c8957
- **Admin.sol**
f336e6bd77e29368a3afe4ffecdc9eafe0b2854f2c303d47405a45a85bfcfb6e
- **ERC721BaseToken.sol**
aab7dd819e3606949889fb0511ebac0c6b9108ffc28503813a3f1dac3a26d230
- **LandBaseToken.sol**
ebb6ab14f7766bc12a1d7c98566160d2ac4350c84c2294ba1d0f5623bcbbca48
- **LandSaleWithETHAndDAI.sol**
95e9422ac6624368ec467c446afb5a4c5cf39f1f210796e314147af96a2058ae
- **MetaTransactionReceiver.sol**
8bae54108e69e81fcffe22425c311814d7339e078ae37e9c1c67c30cf4e4a6e9
- **SafeMathWithRequire.sol**
f7b98afacff77193838a9fdfb4f22457b1734951ec5694249186fd42c5730ff1
- **SuperOperators.sol**
307c0411cfc020057e1d38d9ff5a715b088bd074a8351f1cf572fe2b386dfe12

Summary

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

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

Recommendations

Items in this section are labeled CRITICAL, MAJOR, MINOR, INFO, and DISCUSSION in decreasing significance level.

Admin.sol commit 97013dcbcb29032ea8adba065a9e490138d25713, previous

¹Commit: 752e899abe7d5492227d28470a0bc2a0ae6dfd41

- **INFO** `changeAdmin()`: recommend using the pull over push pattern in case of human errors.
 - (The Sandbox - confirmed): We decided to leave as is as we want to set it to the zero address in the future and we will make sure we do not set it by mistake.

ERC721BaseToken.sol commit 41bd380dfb2c3a88ef23570d2114e9cc8bbf9f, previous

1. **MAJOR** `burnFrom(from, id)`: When item `id`'s operator is set to its owner, `burnFrom` function will enable anyone to burn the item.
 - (The Sandbox - updated): Fixed in commit 752e899abe7d5492227d28470a0bc2a0ae6dfd41.
2. **INFO** `constructor()`: Recommend setting `msg.sender` as the initial admin and change the admin using the pull over push pattern later if it is necessary in case of initial human error.
 - (The Sandbox - confirmed): We disagree as we want to ensure the deployment account's only purpose is to deploy contract. It must not have any other responsibilities.
3. **MINOR** `_transferFrom()`, `_burn()`, `_batchTransferFrom`: Please use `SafeMath` throughout the contract for arithmetic operations.
 - (The Sandbox - confirmed): We consider it is of no use if the logic of the contract ensure it will not happen.
4. **MINOR** `_transferFrom()`: The check for the validity of the transfer should be added to the function before making modifications to states. In the current code, `_checkTransfer()` is called before each call of `_transferFrom()` so the code is safe. However, this pattern is not guaranteed in future implementations, so we recommend adding `_checkTransfer()` inside of `_transferFrom()` or wrapped as modifier.
 - (The Sandbox - confirmed): We will leave as is as we might need to have different logic for checking validity in different implementation.
5. **INFO** `mapping (uint256 => uint256) public _owners`: saving information of `address` owner and `bool` `operatorEnabled` in a `uint256` is of high efficiency. However, this data structure requires developers to stay aware of the changes when they are trying to make conversion between `uint256` and `address`. A separate mapping for checking whether the operator is enabled is recommended as well.
 - (The Sandbox - confirmed): We will leave as is as we think the optimization benefit outweigh the need to ensure it is reset properly.

Land.sol commit 41bd380dfb2c3a88ef23570d2114e9cc8bbf9f, previous

1. **INFO** `uint2str()`: Recommend using `uint256` instead of `uint`. They are exactly the same and `uint` does not bring any problem. However, using `uint256` makes the code readable and consistent.

LandBaseToken.sol commit 41bd380dfb2c3a88ef23570d2114e9cc8bbf9f, previous

1. **INFO** `constructor()`: Recommend setting `msg.sender` as the initial `admin` and change the `admin` using the pull over push pattern later if it is necessary in case of initial human error.
 - (The Sandbox - confirmed): We disagree as we want to ensure the deployment account's only purpose is to deploy contract. It must not have any other responsibilities.
2. **MINOR** Please use `SafeMath` throughout the contract for arithmetic operations.
 - (The Sandbox - confirmed): We consider it is of no use if the logic of the contract ensure it will not happen.
3. **INFO** `width()` and `height()`: Recommend marking these functions with `pure`.
4. **INFO** `x()` and `y()`: Recommend marking these functions with `view`.
5. **INFO** `mintQuad()` `transferQuad()` and `_regroup()`: Recommend checking `size` before, instead of after, using it in order to have a correct error message.
 - (The Sandbox - confirmed): It is currently done just after the coordinates and do not feel like it needs to be changed as coordinates need to be correct anyway.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File LandSale.sol

```
1 pragma solidity 0.5.9;
```

! Version to compile has the following bug: 0.5.9: SignedArrayStorageCopy, ABIEncoderV2StorageArrayV

TIMESTAMP_DEPENDENCY

Line 114 in File LandSale.sol

```
114 require(block.timestamp < _expiryTime, "sale is over");
```

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

INSECURE_COMPILER_VERSION

Line 1 in File SafeMathWithRequire.sol

```
1 pragma solidity ^0.5.2;
```

i Only these compiler versions are safe to compile your code: 0.5.10

INSECURE_COMPILER_VERSION

Line 1 in File LandSaleWithETHAndDAI.sol

```
1 pragma solidity 0.5.9;
```

! Version to compile has the following bug: 0.5.9: SignedArrayStorageCopy, ABIEncoderV2StorageArrayV

INSECURE_COMPILER_VERSION

Line 1 in File Admin.sol

```
1 pragma solidity ^0.5.2;
```

i Only these compiler versions are safe to compile your code: 0.5.10

INSECURE_COMPILER_VERSION

Line 1 in File MetaTransactionReceiver.sol

```
1 pragma solidity ^0.5.2;
```

i Only these compiler versions are safe to compile your code: 0.5.10

INSECURE_COMPILER_VERSION

Line 2 in File ERC721BaseToken.sol

```
2 pragma solidity 0.5.9;
```

! Version to compile has the following bug: 0.5.9: SignedArrayStorageCopy, ABIEncoderV2StorageArrayV

INSECURE_COMPILER_VERSION

Line 1 in File SuperOperators.sol

```
1 pragma solidity ^0.5.2;
```

 Only these compiler versions are safe to compile your code: 0.5.10

INSECURE_COMPILER_VERSION

Line 1 in File AddressUtils.sol

```
1 pragma solidity ^0.5.2;
```

 Only these compiler versions are safe to compile your code: 0.5.10

INSECURE_COMPILER_VERSION

Line 3 in File Land.sol

```
3 pragma solidity 0.5.9;
```

 Version to compile has the following bug: 0.5.9: SignedArrayStorageCopy, ABIEncoderV2StorageArrayV

INSECURE_COMPILER_VERSION

Line 2 in File LandBaseToken.sol

```
2 pragma solidity 0.5.9;
```




 Version to compile has the following bug: 0.5.9: SignedArrayStorageCopy, ABIEncoderV2StorageArrayV

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	 20, Oct 2018
Verification timespan	 395.38 ms
CERTIK label location	Line 30-34 in File howtoread.sol
CERTIK label	<pre> 30 /*@CTK FAIL "transferFrom to same address" 31 @tag assume_completion 32 @pre from == to 33 @post __post.allowed[from][msg.sender] == 34 */ </pre>
Raw code location	Line 35-41 in File howtoread.sol
Raw code	<pre> 35 function transferFrom(address from, address to) { 36 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 } </pre>
Counterexample	 This code violates the specification
Initial environment	<pre> 1 Counter Example: 2 Before Execution: 3 Input = { 4 from = 0x0 5 to = 0x0 6 tokens = 0x6c 7 } 8 This = 0 </pre>
Post environment	<pre> 52 } 53 balance: 0x0 54 } 55 } 56 57 After Execution: 58 Input = { 59 from = 0x0 60 to = 0x0 61 tokens = 0x6c </pre>

Formal Verification Request 1

If method completes, integer overflow would not happen.



10, Dec 2019



34.72 ms

Line 30 in File LandSale.sol

```
30  // @CTK_NO_OVERFLOW
```

Line 43-59 in File LandSale.sol

```
43  constructor(  
44      address landAddress,  
45      address sandContractAddress,  
46      address initialMetaTx,  
47      address admin,  
48      address payable initialWalletAddress,  
49      bytes32 merkleRoot,  
50      uint256 expiryTime  
51  ) public {  
52      _land = Land(landAddress);  
53      _sand = ERC20(sandContractAddress);  
54      _setMetaTransactionProcessor(initialMetaTx, true);  
55      _admin = admin;  
56      _wallet = initialWalletAddress;  
57      _merkleRoot = merkleRoot;  
58      _expiryTime = expiryTime;  
59  }
```



The code meets the specification.

Formal Verification Request 2

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



10, Dec 2019



0.53 ms

Line 31 in File LandSale.sol

```
31  // @CTK_NO_BUF_OVERFLOW
```

Line 43-59 in File LandSale.sol

```
43  constructor(  
44      address landAddress,  
45      address sandContractAddress,  
46      address initialMetaTx,  
47      address admin,  
48      address payable initialWalletAddress,  
49      bytes32 merkleRoot,  
50      uint256 expiryTime  
51  ) public {  
52      _land = Land(landAddress);  
53      _sand = ERC20(sandContractAddress);  
54      _setMetaTransactionProcessor(initialMetaTx, true);  
55      _admin = admin;
```

```

56     _wallet = initialWalletAddress;
57     _merkleRoot = merkleRoot;
58     _expiryTime = expiryTime;
59 }

```

✓ The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.



10, Dec 2019



0.5 ms

Line 32 in File LandSale.sol

```

32  //©CTK NO_ASF

```

Line 43-59 in File LandSale.sol

```

43  constructor(
44      address landAddress,
45      address sandContractAddress,
46      address initialMetaTx,
47      address admin,
48      address payable initialWalletAddress,
49      bytes32 merkleRoot,
50      uint256 expiryTime
51  ) public {
52      _land = Land(landAddress);
53      _sand = ERC20(sandContractAddress);
54      _setMetaTransactionProcessor(initialMetaTx, true);
55      _admin = admin;
56      _wallet = initialWalletAddress;
57      _merkleRoot = merkleRoot;
58      _expiryTime = expiryTime;
59  }

```

✓ The code meets the specification.

Formal Verification Request 4

LandSale



10, Dec 2019



2.44 ms

Line 33-42 in File LandSale.sol

```

33  /*©CTK LandSale
34      @tag assume_completion
35      @post __post._land == landAddress
36      @post __post._sand == sandContractAddress
37      @post __post._metaTransactionContracts[initialMetaTx] == true
38      @post __post._admin == admin
39      @post __post._wallet == initialWalletAddress

```



```
40     @post __post._merkleRoot == merkleRoot
41     @post __post._expiryTime == expiryTime
42     */
```

Line 43-59 in File LandSale.sol

```
43     constructor(
44         address landAddress,
45         address sandContractAddress,
46         address initialMetaTx,
47         address admin,
48         address payable initialWalletAddress,
49         bytes32 merkleRoot,
50         uint256 expiryTime
51     ) public {
52         _land = Land(landAddress);
53         _sand = ERC20(sandContractAddress);
54         _setMetaTransactionProcessor(initialMetaTx, true);
55         _admin = admin;
56         _wallet = initialWalletAddress;
57         _merkleRoot = merkleRoot;
58         _expiryTime = expiryTime;
59     }
```

✓ The code meets the specification.

Formal Verification Request 5

If method completes, integer overflow would not happen.



10, Dec 2019



24.07 ms

Line 63 in File LandSale.sol

```
63     //@CTK NO_OVERFLOW
```

Line 75-79 in File LandSale.sol

```
75     function setReceivingWallet(address payable newWallet) external{
76         require(newWallet != address(0), "receiving wallet cannot be zero address");
77         require(msg.sender == _admin, "only admin can change the receiving wallet");
78         _wallet = newWallet;
79     }
```

✓ The code meets the specification.

Formal Verification Request 6

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



10, Dec 2019



0.52 ms

Line 64 in File LandSale.sol

```
64     //@CTK NO_BUF_OVERFLOW
```

Line 75-79 in File LandSale.sol

```
75     function setReceivingWallet(address payable newWallet) external{
76         require(newWallet != address(0), "receiving wallet cannot be zero address");
77         require(msg.sender == _admin, "only admin can change the receiving wallet");
78         _wallet = newWallet;
79     }
```

✓ The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.39 ms

Line 65 in File LandSale.sol

```
65     //@CTK NO_ASF
```

Line 75-79 in File LandSale.sol

```
75     function setReceivingWallet(address payable newWallet) external{
76         require(newWallet != address(0), "receiving wallet cannot be zero address");
77         require(msg.sender == _admin, "only admin can change the receiving wallet");
78         _wallet = newWallet;
79     }
```

✓ The code meets the specification.

Formal Verification Request 8

setReceivingWallet_require

📅 10, Dec 2019

🕒 2.66 ms

Line 66-70 in File LandSale.sol

```
66     /*@CTK setReceivingWallet_require
67         @tag assume_completion
68         @post newWallet != address(0)
69         @post msg.sender == _admin
70     */
```

Line 75-79 in File LandSale.sol

```
75     function setReceivingWallet(address payable newWallet) external{
76         require(newWallet != address(0), "receiving wallet cannot be zero address");
77         require(msg.sender == _admin, "only admin can change the receiving wallet");
78         _wallet = newWallet;
79     }
```

✓ The code meets the specification.

Formal Verification Request 9

setReceivingWallet_change



10, Dec 2019



2.44 ms

Line 71-74 in File LandSale.sol

```

71  /*@CTK setReceivingWallet_change
72    @tag assume_completion
73    @post __post._wallet == newWallet
74  */

```

Line 75-79 in File LandSale.sol

```

75  function setReceivingWallet(address payable newWallet) external{
76    require(newWallet != address(0), "receiving wallet cannot be zero address");
77    require(msg.sender == _admin, "only admin can change the receiving wallet");
78    _wallet = newWallet;
79  }

```

✓ The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.



10, Dec 2019



131.38 ms

Line 93 in File LandSale.sol

```

93  //@CTK NO_OVERFLOW

```

Line 102-137 in File LandSale.sol

```

102  function buyLandWithSand(
103    address buyer,
104    address to,
105    address reserved,
106    uint256 x,
107    uint256 y,
108    uint256 size,
109    uint256 price,
110    bytes32 salt,
111    bytes32[] calldata proof
112  ) external {
113    /* solhint-disable-next-line not-rely-on-time */
114    require(block.timestamp < _expiryTime, "sale is over");
115    require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
      authorized");
116    require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
117    bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
118
119    require(
120      _verify(proof, leaf),
121      "Invalid land provided"
122    );

```

```

123
124     require(
125         _sand.transferFrom(
126             buyer,
127             _wallet,
128             price
129         ),
130         "sand transfer failed"
131     );
132
133     _land.mintQuad(to, size, x, y, "");
134     emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);
135 }


```

✓ The code meets the specification.

Formal Verification Request 11

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

 10, Dec 2019

 15.94 ms

Line 94 in File LandSale.sol

```
94 // @CTK NO_BUF_OVERFLOW
```

Line 102-137 in File LandSale.sol

```

102     function buyLandWithSand(
103         address buyer,
104         address to,
105         address reserved,
106         uint256 x,
107         uint256 y,
108         uint256 size,
109         uint256 price,
110         bytes32 salt,
111         bytes32[] calldata proof
112     ) external {
113         /* solhint-disable-next-line not-rely-on-time */
114         require(block.timestamp < _expiryTime, "sale is over");
115         require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
            authorized");
116         require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
117         bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
118
119         require(
120             _verify(proof, leaf),
121             "Invalid land provided"
122         );
123
124         require(
125             _sand.transferFrom(
126                 buyer,
127                 _wallet,
128                 price
129             ),

```

```

130         "sand transfer failed"
131     );
132
133     _land.mintQuad(to, size, x, y, "");
134     emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);
135 }

```

✓ The code meets the specification.

Formal Verification Request 12

Method will not encounter an assertion failure.



10, Dec 2019



16.58 ms

Line 95 in File LandSale.sol

```
95 //CTK NO_ASF
```

Line 102-137 in File LandSale.sol

```

102 function buyLandWithSand(
103     address buyer,
104     address to,
105     address reserved,
106     uint256 x,
107     uint256 y,
108     uint256 size,
109     uint256 price,
110     bytes32 salt,
111     bytes32[] calldata proof
112 ) external {
113     /* solhint-disable-next-line not-rely-on-time */
114     require(block.timestamp < _expiryTime, "sale is over");
115     require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
        authorized");
116     require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
117     bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
118
119     require(
120         _verify(proof, leaf),
121         "Invalid land provided"
122     );
123
124     require(
125         _sand.transferFrom(
126             buyer,
127             _wallet,
128             price
129         ),
130         "sand transfer failed"
131     );
132
133     _land.mintQuad(to, size, x, y, "");
134     emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);
135 }

```

✓ The code meets the specification.

Formal Verification Request 13

buyLandWithSand

📅 10, Dec 2019

🕒 16.74 ms

Line 96-101 in File LandSale.sol

```

96  /*@CTK buyLandWithSand
97    @tag assume_completion
98    @pre _expiryTime > block.timestamp
99    @post buyer == msg.sender /\ _metaTransactionContracts[msg.sender] == true
100    @post reserved == address(0) /\ reserved == buyer
101  */

```

Line 102-137 in File LandSale.sol

```

102  function buyLandWithSand(
103      address buyer,
104      address to,
105      address reserved,
106      uint256 x,
107      uint256 y,
108      uint256 size,
109      uint256 price,
110      bytes32 salt,
111      bytes32[] calldata proof
112  ) external {
113      /* solhint-disable-next-line not-rely-on-time */
114      require(block.timestamp < _expiryTime, "sale is over");
115      require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
116              authorized");
117      require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
118      bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
119
120      require(
121          _verify(proof, leaf),
122          "Invalid land provided"
123      );
124
125      require(
126          _sand.transferFrom(
127              buyer,
128              _wallet,
129              price
130          ),
131          "sand transfer failed"
132      );
133
134      _land.mintQuad(to, size, x, y, "");
135      emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);

```

✓ The code meets the specification.

Formal Verification Request 14

getExpiryTime



10, Dec 2019



4.23 ms

Line 143-145 in File LandSale.sol

```
143  /*@CTK getExpiryTime
144    @post __return == _expiryTime
145  */
```

Line 146-148 in File LandSale.sol

```
146  function getExpiryTime() external view returns(uint256) {
147    return _expiryTime;
148  }
```



The code meets the specification.

Formal Verification Request 15

merkleRoot



10, Dec 2019



4.22 ms

Line 154-156 in File LandSale.sol

```
154  /*@CTK merkleRoot
155    @post __return == _merkleRoot
156  */
```

Line 157-159 in File LandSale.sol

```
157  function merkleRoot() external view returns(bytes32) {
158    return _merkleRoot;
159  }
```



The code meets the specification.

Formal Verification Request 16

If method completes, integer overflow would not happen.



10, Dec 2019



0.29 ms

Line 161 in File LandSale.sol

```
161  //@CTK NO_OVERFLOW
```

Line 164-184 in File LandSale.sol

```

164 function _generateLandHash(
165     uint256 x,
166     uint256 y,
167     uint256 size,
168     uint256 price,
169     address reserved,
170     bytes32 salt
171 ) internal pure returns (
172     bytes32
173 ) {
174     return keccak256(
175         abi.encodePacked(
176             x,
177             y,
178             size,
179             price,
180             reserved,
181             salt
182         )
183     );
184 }

```

✓ The code meets the specification.

Formal Verification Request 17

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



10, Dec 2019



0.28 ms

Line 162 in File LandSale.sol

```

162 // @CTK NO_BUF_OVERFLOW

```

Line 164-184 in File LandSale.sol

```

164 function _generateLandHash(
165     uint256 x,
166     uint256 y,
167     uint256 size,
168     uint256 price,
169     address reserved,
170     bytes32 salt
171 ) internal pure returns (
172     bytes32
173 ) {
174     return keccak256(
175         abi.encodePacked(
176             x,
177             y,
178             size,
179             price,
180             reserved,
181             salt
182         )
183     );
184 }

```


✓ The code meets the specification.

Formal Verification Request 18

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.28 ms

Line 163 in File LandSale.sol

163 `//@CTK NO_ASF`

Line 164-184 in File LandSale.sol

```
164     function _generateLandHash(  
165         uint256 x,  
166         uint256 y,  
167         uint256 size,  
168         uint256 price,  
169         address reserved,  
170         bytes32 salt  
171     ) internal pure returns (  
172         bytes32  
173     ) {  
174         return keccak256(  
175             abi.encodePacked(  
176                 x,  
177                 y,  
178                 size,  
179                 price,  
180                 reserved,  
181                 salt  
182             )  
183         );  
184     }
```

✓ The code meets the specification.

Formal Verification Request 19

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 0.46 ms

Line 186 in File LandSale.sol

186 `//@CTK NO_OVERFLOW`

Line 189-209 in File LandSale.sol

```
189     function _verify(bytes32[] memory proof, bytes32 leaf) internal view returns (bool) {  
190         bytes32 computedHash = leaf;  
191  
192         /*@CTK _verify_loop  
193         @inv i <= proof.length
```

```

194     @post i == proof.length
195     */
196     for (uint256 i = 0; i < proof.length; i++) {
197         bytes32 proofElement = proof[i];
198
199         if (computedHash < proofElement) {
200             computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
201         } else {
202             computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
203         }
204     }
205
206     return computedHash == _merkleRoot;
207 }

```

✓ The code meets the specification.

Formal Verification Request 20

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

📅 10, Dec 2019

🕒 0.31 ms

Line 187 in File LandSale.sol

```

187     //@CTK NO_BUF_OVERFLOW

```

Line 189-209 in File LandSale.sol

```

189     function _verify(bytes32[] memory proof, bytes32 leaf) internal view returns (bool) {
190         bytes32 computedHash = leaf;
191
192         /*@CTK _verify_loop
193         @inv i <= proof.length
194         @post i == proof.length
195         */
196         for (uint256 i = 0; i < proof.length; i++) {
197             bytes32 proofElement = proof[i];
198
199             if (computedHash < proofElement) {
200                 computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
201             } else {
202                 computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
203             }
204         }
205
206         return computedHash == _merkleRoot;
207     }

```

✓ The code meets the specification.

Formal Verification Request 21

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.31 ms

Line 188 in File LandSale.sol

188 //CTK NO_ASF

Line 189-209 in File LandSale.sol

```

189 function _verify(bytes32[] memory proof, bytes32 leaf) internal view returns (bool) {
190     bytes32 computedHash = leaf;
191
192     /*CTK _verify_loop
193         @inv i <= proof.length
194         @post i == proof.length
195     */
196     for (uint256 i = 0; i < proof.length; i++) {
197         bytes32 proofElement = proof[i];
198
199         if (computedHash < proofElement) {
200             computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
201         } else {
202             computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
203         }
204     }
205
206     return computedHash == _merkleRoot;
207 }

```

✅ The code meets the specification.

Formal Verification Request 22

__verify_loop__ Generated

📅 10, Dec 2019

🕒 17.06 ms

(Loop) Line 192-195 in File LandSale.sol

```

192 /*CTK _verify_loop
193     @inv i <= proof.length
194     @post i == proof.length
195 */

```

(Loop) Line 192-206 in File LandSale.sol

```

192 /*CTK _verify_loop
193     @inv i <= proof.length
194     @post i == proof.length
195 */
196 for (uint256 i = 0; i < proof.length; i++) {
197     bytes32 proofElement = proof[i];
198
199     if (computedHash < proofElement) {
200         computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
201     } else {
202         computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
203     }
204 }

```

✓ The code meets the specification.

Formal Verification Request 23

If method completes, integer overflow would not happen.



10, Dec 2019



34.37 ms

Line 11 in File SafeMathWithRequire.sol

```
11 //@CTK NO_OVERFLOW
```

Line 18-29 in File SafeMathWithRequire.sol

```
18 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
19     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
20     // benefit is lost if 'b' is also tested.
21     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
22     if (a == 0) {
23         return 0;
24     }
25
26     uint256 c = a * b;
27     require(c / a == b, "overflow");
28     return c;
29 }
```

✓ The code meets the specification.

Formal Verification Request 24

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



10, Dec 2019



0.59 ms

Line 12 in File SafeMathWithRequire.sol

```
12 //@CTK NO_BUF_OVERFLOW
```

Line 18-29 in File SafeMathWithRequire.sol

```
18 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
19     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
20     // benefit is lost if 'b' is also tested.
21     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
22     if (a == 0) {
23         return 0;
24     }
25
26     uint256 c = a * b;
27     require(c / a == b, "overflow");
28     return c;
29 }
```

✓ The code meets the specification.

Formal Verification Request 25

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.49 ms

Line 13 in File SafeMathWithRequire.sol

```
13 // @CTK NO_ASF
```

Line 18-29 in File SafeMathWithRequire.sol

```
18 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
19     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
20     // benefit is lost if 'b' is also tested.
21     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
22     if (a == 0) {
23         return 0;
24     }
25
26     uint256 c = a * b;
27     require(c / a == b, "overflow");
28     return c;
29 }
```

✅ The code meets the specification.

Formal Verification Request 26

mul

📅 10, Dec 2019

🕒 2.94 ms

Line 14-17 in File SafeMathWithRequire.sol

```
14 /* @CTK mul
15    @tag assume_completion
16    @post __return == a * b
17 */
```

Line 18-29 in File SafeMathWithRequire.sol

```
18 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
19     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
20     // benefit is lost if 'b' is also tested.
21     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
22     if (a == 0) {
23         return 0;
24     }
25
26     uint256 c = a * b;
27     require(c / a == b, "overflow");
28     return c;
29 }
```

✅ The code meets the specification.

Formal Verification Request 27

If method completes, integer overflow would not happen.



10, Dec 2019



5.46 ms

Line 34 in File SafeMathWithRequire.sol

```
34  // @CTK_NO_OVERFLOW
```

Line 41-46 in File SafeMathWithRequire.sol

```
41  function div(uint256 a, uint256 b) internal pure returns (uint256) {
42      // assert(b > 0); // Solidity automatically throws when dividing by 0
43      // uint256 c = a / b;
44      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45      return a / b;
46  }
```



The code meets the specification.

Formal Verification Request 28

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



10, Dec 2019



0.41 ms

Line 35 in File SafeMathWithRequire.sol

```
35  // @CTK_NO_BUF_OVERFLOW
```

Line 41-46 in File SafeMathWithRequire.sol

```
41  function div(uint256 a, uint256 b) internal pure returns (uint256) {
42      // assert(b > 0); // Solidity automatically throws when dividing by 0
43      // uint256 c = a / b;
44      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45      return a / b;
46  }
```



The code meets the specification.

Formal Verification Request 29

Method will not encounter an assertion failure.



10, Dec 2019



0.44 ms

Line 36 in File SafeMathWithRequire.sol

```
36  // @CTK_FAIL_NO_ASF
```

Line 41-46 in File SafeMathWithRequire.sol

```

41 function div(uint256 a, uint256 b) internal pure returns (uint256) {
42     // assert(b > 0); // Solidity automatically throws when dividing by 0
43     // uint256 c = a / b;
44     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45     return a / b;
46 }

```

✖ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     a = 0
5     b = 0
6   }
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    __return = 0
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25  }
26  Address_Map = [
27    {
28      "key": "ALL_OTHERS",
29      "value": "EmptyAddress"
30    }
31  ]
32
33 Function invocation is reverted.

```

Formal Verification Request 30

div



10, Dec 2019



0.34 ms

Line 37-40 in File SafeMathWithRequire.sol

```

37 /*@CTK div
38   @tag assume_completion
39   @post __return == a / b
40 */

```

Line 41-46 in File SafeMathWithRequire.sol

```
41 function div(uint256 a, uint256 b) internal pure returns (uint256) {
42     // assert(b > 0); // Solidity automatically throws when dividing by 0
43     // uint256 c = a / b;
44     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45     return a / b;
46 }
```

✓ The code meets the specification.

Formal Verification Request 31

If method completes, integer overflow would not happen.



10, Dec 2019



10.86 ms

Line 51 in File SafeMathWithRequire.sol

```
51 // @CTK_NO_OVERFLOW
```

Line 58-61 in File SafeMathWithRequire.sol

```
58 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
59     require(b <= a, "undeflow");
60     return a - b;
61 }
```

✓ The code meets the specification.

Formal Verification Request 32

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



10, Dec 2019



0.41 ms

Line 52 in File SafeMathWithRequire.sol

```
52 // @CTK_NO_BUF_OVERFLOW
```

Line 58-61 in File SafeMathWithRequire.sol

```
58 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
59     require(b <= a, "undeflow");
60     return a - b;
61 }
```

✓ The code meets the specification.

Formal Verification Request 33

Method will not encounter an assertion failure.



10, Dec 2019



0.42 ms

Line 53 in File SafeMathWithRequire.sol


```
53 // @CTK NO_ASF
```

Line 58-61 in File SafeMathWithRequire.sol

```
58 function sub(uint256 a, uint256 b) internal pure returns (uint256) {  
59     require(b <= a, "undeflow");  
60     return a - b;  
61 }
```

✓ The code meets the specification.

Formal Verification Request 34

sub

📅 10, Dec 2019

🕒 0.75 ms

Line 54-57 in File SafeMathWithRequire.sol

```
54 /* @CTK sub  
55     @tag assume_completion  
56     @post __return == a - b  
57 */
```

Line 58-61 in File SafeMathWithRequire.sol

```
58 function sub(uint256 a, uint256 b) internal pure returns (uint256) {  
59     require(b <= a, "undeflow");  
60     return a - b;  
61 }
```

✓ The code meets the specification.

Formal Verification Request 35

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 13.53 ms

Line 66 in File SafeMathWithRequire.sol

```
66 // @CTK NO_OVERFLOW
```

Line 73-77 in File SafeMathWithRequire.sol

```
73 function add(uint256 a, uint256 b) internal pure returns (uint256) {  
74     uint256 c = a + b;  
75     require(c >= a, "overflow");  
76     return c;  
77 }
```

✓ The code meets the specification.

Formal Verification Request 36

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

10, Dec 2019

0.35 ms

Line 67 in File SafeMathWithRequire.sol

```
67 // @CTK_NO_BUF_OVERFLOW
```

Line 73-77 in File SafeMathWithRequire.sol

```
73 function add(uint256 a, uint256 b) internal pure returns (uint256) {  
74     uint256 c = a + b;  
75     require(c >= a, "overflow");  
76     return c;  
77 }
```

✓ The code meets the specification.

Formal Verification Request 37

Method will not encounter an assertion failure.

10, Dec 2019

0.34 ms

Line 68 in File SafeMathWithRequire.sol

```
68 // @CTK_NO_ASF
```

Line 73-77 in File SafeMathWithRequire.sol

```
73 function add(uint256 a, uint256 b) internal pure returns (uint256) {  
74     uint256 c = a + b;  
75     require(c >= a, "overflow");  
76     return c;  
77 }
```

✓ The code meets the specification.

Formal Verification Request 38

add

10, Dec 2019

0.68 ms

Line 69-72 in File SafeMathWithRequire.sol

```
69 /* @CTK add  
70     @tag assume_completion  
71     @post __return == a + b  
72 */
```

Line 73-77 in File SafeMathWithRequire.sol

```

73     function add(uint256 a, uint256 b) internal pure returns (uint256) {
74         uint256 c = a + b;
75         require(c >= a, "overflow");
76         return c;
77     }

```

✓ The code meets the specification.

Formal Verification Request 39

If method completes, integer overflow would not happen.



10, Dec 2019



53.57 ms

Line 42 in File LandSaleWithETHAndDAI.sol

```

42     //@CTK NO_OVERFLOW

```

Line 57-77 in File LandSaleWithETHAndDAI.sol

```

57     constructor(
58         address landAddress,
59         address sandContractAddress,
60         address initialMetaTx,
61         address admin,
62         address payable initialWalletAddress,
63         bytes32 merkleRoot,
64         uint256 expiryTime,
65         address medianizerContractAddress,
66         address daiTokenContractAddress
67     ) public {
68         _land = Land(landAddress);
69         _sand = ERC20(sandContractAddress);
70         _setMetaTransactionProcessor(initialMetaTx, true);
71         _admin = admin;
72         _wallet = initialWalletAddress;
73         _merkleRoot = merkleRoot;
74         _expiryTime = expiryTime;
75         _medianizer = Medianizer(medianizerContractAddress);
76         _dai = ERC20(daiTokenContractAddress);
77     }

```

✓ The code meets the specification.

Formal Verification Request 40

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



10, Dec 2019



0.53 ms

Line 43 in File LandSaleWithETHAndDAI.sol

```

43     //@CTK NO_BUF_OVERFLOW

```

Line 57-77 in File LandSaleWithETHAndDAI.sol

```

57     constructor(
58         address landAddress,
59         address sandContractAddress,
60         address initialMetaTx,
61         address admin,
62         address payable initialWalletAddress,
63         bytes32 merkleRoot,
64         uint256 expiryTime,
65         address medianizerContractAddress,
66         address daiTokenContractAddress
67     ) public {
68         _land = Land(landAddress);
69         _sand = ERC20(sandContractAddress);
70         _setMetaTransactionProcessor(initialMetaTx, true);
71         _admin = admin;
72         _wallet = initialWalletAddress;
73         _merkleRoot = merkleRoot;
74         _expiryTime = expiryTime;
75         _medianizer = Medianizer(medianizerContractAddress);
76         _dai = ERC20(daiTokenContractAddress);
77     }

```

✓ The code meets the specification.

Formal Verification Request 41

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.55 ms

Line 44 in File LandSaleWithETHAndDAI.sol

```
44 //©CTK NO_ASF
```

Line 57-77 in File LandSaleWithETHAndDAI.sol

```

57     constructor(
58         address landAddress,
59         address sandContractAddress,
60         address initialMetaTx,
61         address admin,
62         address payable initialWalletAddress,
63         bytes32 merkleRoot,
64         uint256 expiryTime,
65         address medianizerContractAddress,
66         address daiTokenContractAddress
67     ) public {
68         _land = Land(landAddress);
69         _sand = ERC20(sandContractAddress);
70         _setMetaTransactionProcessor(initialMetaTx, true);
71         _admin = admin;
72         _wallet = initialWalletAddress;
73         _merkleRoot = merkleRoot;
74         _expiryTime = expiryTime;
75         _medianizer = Medianizer(medianizerContractAddress);
76         _dai = ERC20(daiTokenContractAddress);
77     }

```

✓ The code meets the specification.

Formal Verification Request 42

LandSale

📅 10, Dec 2019

🕒 3.8 ms

Line 45-56 in File LandSaleWithETHAndDAI.sol

```
45  /*@CTK LandSale
46  @tag assume_completion
47  @post __post._land == landAddress
48  @post __post._sand == sandContractAddress
49  @post __post._metaTransactionContracts[initialMetaTx] == true
50  @post __post._admin == admin
51  @post __post._wallet == initialWalletAddress
52  @post __post._merkleRoot == merkleRoot
53  @post __post._expiryTime == expiryTime
54  @post __post._medianizer == medianizerContractAddress
55  @post __post._dai == daiTokenContractAddress
56  */
```

Line 57-77 in File LandSaleWithETHAndDAI.sol

```
57  constructor(
58      address landAddress,
59      address sandContractAddress,
60      address initialMetaTx,
61      address admin,
62      address payable initialWalletAddress,
63      bytes32 merkleRoot,
64      uint256 expiryTime,
65      address medianizerContractAddress,
66      address daiTokenContractAddress
67  ) public {
68      _land = Land(landAddress);
69      _sand = ERC20(sandContractAddress);
70      _setMetaTransactionProcessor(initialMetaTx, true);
71      _admin = admin;
72      _wallet = initialWalletAddress;
73      _merkleRoot = merkleRoot;
74      _expiryTime = expiryTime;
75      _medianizer = Medianizer(medianizerContractAddress);
76      _dai = ERC20(daiTokenContractAddress);
77  }
```

✓ The code meets the specification.

Formal Verification Request 43

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 23.15 ms

Line 81 in File LandSaleWithETHAndDAI.sol

```
81 // @CTK_NO_OVERFLOW
```

Line 93-97 in File LandSaleWithETHAndDAI.sol

```
93 function setReceivingWallet(address payable newWallet) external{
94     require(newWallet != address(0), "receiving wallet cannot be zero address");
95     require(msg.sender == _admin, "only admin can change the receiving wallet");
96     _wallet = newWallet;
97 }
```

✓ The code meets the specification.

Formal Verification Request 44

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

📅 10, Dec 2019

🕒 0.38 ms

Line 82 in File LandSaleWithETHAndDAI.sol

```
82 // @CTK_NO_BUF_OVERFLOW
```

Line 93-97 in File LandSaleWithETHAndDAI.sol

```
93 function setReceivingWallet(address payable newWallet) external{
94     require(newWallet != address(0), "receiving wallet cannot be zero address");
95     require(msg.sender == _admin, "only admin can change the receiving wallet");
96     _wallet = newWallet;
97 }
```

✓ The code meets the specification.

Formal Verification Request 45

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.37 ms

Line 83 in File LandSaleWithETHAndDAI.sol

```
83 // @CTK_NO_ASF
```

Line 93-97 in File LandSaleWithETHAndDAI.sol

```
93 function setReceivingWallet(address payable newWallet) external{
94     require(newWallet != address(0), "receiving wallet cannot be zero address");
95     require(msg.sender == _admin, "only admin can change the receiving wallet");
96     _wallet = newWallet;
97 }
```

✓ The code meets the specification.

Formal Verification Request 46

setReceivingWallet__require



10, Dec 2019



2.33 ms

Line 84-88 in File LandSaleWithETHAndDAI.sol

```
84      /*CTK setReceivingWallet_require
85         @tag assume_completion
86         @post newWallet != address(0)
87         @post msg.sender == _admin
88      */
```

Line 93-97 in File LandSaleWithETHAndDAI.sol

```
93      function setReceivingWallet(address payable newWallet) external{
94          require(newWallet != address(0), "receiving wallet cannot be zero address");
95          require(msg.sender == _admin, "only admin can change the receiving wallet");
96          _wallet = newWallet;
97      }
```

✓ The code meets the specification.

Formal Verification Request 47

setReceivingWallet__change



10, Dec 2019



1.98 ms

Line 89-92 in File LandSaleWithETHAndDAI.sol

```
89      /*CTK setReceivingWallet_change
90         @tag assume_completion
91         @post __post._wallet == newWallet
92      */
```

Line 93-97 in File LandSaleWithETHAndDAI.sol

```
93      function setReceivingWallet(address payable newWallet) external{
94          require(newWallet != address(0), "receiving wallet cannot be zero address");
95          require(msg.sender == _admin, "only admin can change the receiving wallet");
96          _wallet = newWallet;
97      }
```

✓ The code meets the specification.

Formal Verification Request 48

If method completes, integer overflow would not happen.



10, Dec 2019



14.36 ms

Line 101 in File LandSaleWithETHAndDAI.sol

101 // @CTK_NO_OVERFLOW

Line 112-115 in File LandSaleWithETHAndDAI.sol

```
112 function setDAIEnabled(bool enabled) external {
113     require(msg.sender == _admin, "only admin can enable/disable DAI");
114     _daiEnabled = enabled;
115 }
```

✓ The code meets the specification.

Formal Verification Request 49

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

📅 10, Dec 2019

🕒 0.47 ms

Line 102 in File LandSaleWithETHAndDAI.sol

102 // @CTK_NO_BUF_OVERFLOW

Line 112-115 in File LandSaleWithETHAndDAI.sol

```
112 function setDAIEnabled(bool enabled) external {
113     require(msg.sender == _admin, "only admin can enable/disable DAI");
114     _daiEnabled = enabled;
115 }
```

✓ The code meets the specification.

Formal Verification Request 50

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.63 ms

Line 103 in File LandSaleWithETHAndDAI.sol

103 // @CTK_NO_ASF

Line 112-115 in File LandSaleWithETHAndDAI.sol

```
112 function setDAIEnabled(bool enabled) external {
113     require(msg.sender == _admin, "only admin can enable/disable DAI");
114     _daiEnabled = enabled;
115 }
```

✓ The code meets the specification.

Formal Verification Request 51

setDAIEnabled__require



10, Dec 2019



1.02 ms

Line 104-107 in File LandSaleWithETHAndDAI.sol

```
104  /*@CTK setDAIEnabled__require
105      @tag assume_completion
106      @post msg.sender == _admin
107  */
```

Line 112-115 in File LandSaleWithETHAndDAI.sol

```
112  function setDAIEnabled(bool enabled) external {
113      require(msg.sender == _admin, "only admin can enable/disable DAI");
114      _daiEnabled = enabled;
115  }
```

✓ The code meets the specification.

Formal Verification Request 52

setDAIEnabled__change



10, Dec 2019



2.06 ms

Line 108-111 in File LandSaleWithETHAndDAI.sol

```
108  /*@CTK setDAIEnabled__change
109      @tag assume_completion
110      @post __post._daiEnabled == enabled
111  */
```

Line 112-115 in File LandSaleWithETHAndDAI.sol

```
112  function setDAIEnabled(bool enabled) external {
113      require(msg.sender == _admin, "only admin can enable/disable DAI");
114      _daiEnabled = enabled;
115  }
```

✓ The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.



10, Dec 2019



4.37 ms

Line 119 in File LandSaleWithETHAndDAI.sol

```
119  //@CTK NO_OVERFLOW
```

Line 125-127 in File LandSaleWithETHAndDAI.sol

```
125     function isDAIEnabled() external view returns (bool) {
126         return _daiEnabled;
127     }
```

✓ The code meets the specification.

Formal Verification Request 54

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

📅 10, Dec 2019

🕒 0.51 ms

Line 120 in File LandSaleWithETHAndDAI.sol

```
120     //@CTK_NO_BUF_OVERFLOW
```

Line 125-127 in File LandSaleWithETHAndDAI.sol

```
125     function isDAIEnabled() external view returns (bool) {
126         return _daiEnabled;
127     }
```

✓ The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.42 ms

Line 121 in File LandSaleWithETHAndDAI.sol

```
121     //@CTK_NO_ASF
```

Line 125-127 in File LandSaleWithETHAndDAI.sol

```
125     function isDAIEnabled() external view returns (bool) {
126         return _daiEnabled;
127     }
```

✓ The code meets the specification.

Formal Verification Request 56

isDAIEnabled

📅 10, Dec 2019

🕒 0.32 ms

Line 122-124 in File LandSaleWithETHAndDAI.sol

```
122     /*@CTK isDAIEnabled
123         @post __return == _daiEnabled
124     */
```

Line 125-127 in File LandSaleWithETHAndDAI.sol

```
125     function isDAIEnabled() external view returns (bool) {
126         return _daiEnabled;
127     }
```

✓ The code meets the specification.

Formal Verification Request 57

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 14.37 ms

Line 131 in File LandSaleWithETHAndDAI.sol

```
131     //@CTK NO_OVERFLOW
```

Line 142-145 in File LandSaleWithETHAndDAI.sol

```
142     function setETHEnabled(bool enabled) external {
143         require(msg.sender == _admin, "only admin can enable/disable ETH");
144         _etherEnabled = enabled;
145     }
```

✓ The code meets the specification.

Formal Verification Request 58

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

📅 10, Dec 2019

🕒 0.35 ms

Line 132 in File LandSaleWithETHAndDAI.sol

```
132     //@CTK NO_BUF_OVERFLOW
```

Line 142-145 in File LandSaleWithETHAndDAI.sol

```
142     function setETHEnabled(bool enabled) external {
143         require(msg.sender == _admin, "only admin can enable/disable ETH");
144         _etherEnabled = enabled;
145     }
```

✓ The code meets the specification.

Formal Verification Request 59

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.34 ms

Line 133 in File LandSaleWithETHAndDAI.sol

133 //CTK NO_ASF

Line 142-145 in File LandSaleWithETHAndDAI.sol

```
142 function setETHEnabled(bool enabled) external {
143     require(msg.sender == _admin, "only admin can enable/disable ETH");
144     _etherEnabled = enabled;
145 }
```

✓ The code meets the specification.

Formal Verification Request 60

setETHEnabled_require

📅 10, Dec 2019

🕒 0.78 ms

Line 134-137 in File LandSaleWithETHAndDAI.sol

```
134 /*CTK setETHEnabled_require
135     @tag assume_completion
136     @post msg.sender == _admin
137 */
```

Line 142-145 in File LandSaleWithETHAndDAI.sol

```
142 function setETHEnabled(bool enabled) external {
143     require(msg.sender == _admin, "only admin can enable/disable ETH");
144     _etherEnabled = enabled;
145 }
```

✓ The code meets the specification.

Formal Verification Request 61

setETHEnabled_change

📅 10, Dec 2019

🕒 2.0 ms

Line 138-141 in File LandSaleWithETHAndDAI.sol

```
138 /*CTK setETHEnabled_change
139     @tag assume_completion
140     @post __post._etherEnabled == enabled
141 */
```

Line 142-145 in File LandSaleWithETHAndDAI.sol

```
142 function setETHEnabled(bool enabled) external {
143     require(msg.sender == _admin, "only admin can enable/disable ETH");
144     _etherEnabled = enabled;
145 }
```

✓ The code meets the specification.

Formal Verification Request 62

If method completes, integer overflow would not happen.



10, Dec 2019



4.15 ms

Line 149 in File LandSaleWithETHAndDAI.sol

```
149  // @CTK_NO_OVERFLOW
```

Line 155-157 in File LandSaleWithETHAndDAI.sol

```
155  function isETHEnabled() external view returns (bool) {  
156      return _etherEnabled;  
157  }
```

✓ The code meets the specification.

Formal Verification Request 63

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



10, Dec 2019



0.3 ms

Line 150 in File LandSaleWithETHAndDAI.sol

```
150  // @CTK_NO_BUF_OVERFLOW
```

Line 155-157 in File LandSaleWithETHAndDAI.sol

```
155  function isETHEnabled() external view returns (bool) {  
156      return _etherEnabled;  
157  }
```

✓ The code meets the specification.

Formal Verification Request 64

Method will not encounter an assertion failure.



10, Dec 2019



0.26 ms

Line 151 in File LandSaleWithETHAndDAI.sol

```
151  // @CTK_NO_ASF
```

Line 155-157 in File LandSaleWithETHAndDAI.sol

```
155  function isETHEnabled() external view returns (bool) {  
156      return _etherEnabled;  
157  }
```

✓ The code meets the specification.

Formal Verification Request 65

isETHEnabled



10, Dec 2019



0.27 ms

Line 152-154 in File LandSaleWithETHAndDAI.sol

```
152  /*@CTK isETHEnabled
153     @post __return == _etherEnabled
154  */
```

Line 155-157 in File LandSaleWithETHAndDAI.sol

```
155  function isETHEnabled() external view returns (bool) {
156      return _etherEnabled;
157  }
```

✓ The code meets the specification.

Formal Verification Request 66

If method completes, integer overflow would not happen.



10, Dec 2019



15.02 ms

Line 161 in File LandSaleWithETHAndDAI.sol

```
161  //@CTK NO_OVERFLOW
```

Line 172-175 in File LandSaleWithETHAndDAI.sol

```
172  function setSANDEnabled(bool enabled) external {
173      require(msg.sender == _admin, "only admin can enable/disable SAND");
174      _sandEnabled = enabled;
175  }
```

✓ The code meets the specification.

Formal Verification Request 67

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



10, Dec 2019



0.34 ms

Line 162 in File LandSaleWithETHAndDAI.sol

```
162  //@CTK NO_BUF_OVERFLOW
```

Line 172-175 in File LandSaleWithETHAndDAI.sol

```
172  function setSANDEnabled(bool enabled) external {
173      require(msg.sender == _admin, "only admin can enable/disable SAND");
174      _sandEnabled = enabled;
175  }
```

✓ The code meets the specification.

Formal Verification Request 68

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.33 ms

Line 163 in File LandSaleWithETHAndDAI.sol

```
163 // @CTK NO_ASF
```

Line 172-175 in File LandSaleWithETHAndDAI.sol

```
172 function setSANDEnabled(bool enabled) external {
173     require(msg.sender == _admin, "only admin can enable/disable SAND");
174     _sandEnabled = enabled;
175 }
```

✅ The code meets the specification.

Formal Verification Request 69

setSANDEnabled_require

📅 10, Dec 2019

🕒 0.76 ms

Line 164-167 in File LandSaleWithETHAndDAI.sol

```
164 /* @CTK setSANDEnabled_require
165     @tag assume_completion
166     @post msg.sender == _admin
167 */
```

Line 172-175 in File LandSaleWithETHAndDAI.sol

```
172 function setSANDEnabled(bool enabled) external {
173     require(msg.sender == _admin, "only admin can enable/disable SAND");
174     _sandEnabled = enabled;
175 }
```

✅ The code meets the specification.

Formal Verification Request 70

setSANDEnabled_change

📅 10, Dec 2019

🕒 1.91 ms

Line 168-171 in File LandSaleWithETHAndDAI.sol

```
168 /* @CTK setSANDEnabled_change
169     @tag assume_completion
170     @post __post._sandEnabled == enabled
171 */
```

Line 172-175 in File LandSaleWithETHAndDAI.sol

```
172 function setSANDEnabled(bool enabled) external {
173     require(msg.sender == _admin, "only admin can enable/disable SAND");
174     _sandEnabled = enabled;
175 }
```

✓ The code meets the specification.

Formal Verification Request 71

If method completes, integer overflow would not happen.



10, Dec 2019



3.99 ms

Line 179 in File LandSaleWithETHAndDAI.sol

```
179 // @CTK_NO_OVERFLOW
```

Line 185-187 in File LandSaleWithETHAndDAI.sol

```
185 function isSANDEnabled() external view returns (bool) {
186     return _sandEnabled;
187 }
```

✓ The code meets the specification.

Formal Verification Request 72

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



10, Dec 2019



0.26 ms

Line 180 in File LandSaleWithETHAndDAI.sol

```
180 // @CTK_NO_BUF_OVERFLOW
```

Line 185-187 in File LandSaleWithETHAndDAI.sol

```
185 function isSANDEnabled() external view returns (bool) {
186     return _sandEnabled;
187 }
```

✓ The code meets the specification.

Formal Verification Request 73

Method will not encounter an assertion failure.



10, Dec 2019



0.26 ms

Line 181 in File LandSaleWithETHAndDAI.sol

```
181 // @CTK_NO_ASF
```


Line 185-187 in File LandSaleWithETHAndDAI.sol

```
185     function isSANDEnabled() external view returns (bool) {
186         return _sandEnabled;
187     }
```

✓ The code meets the specification.

Formal Verification Request 74

isSANDEnabled



10, Dec 2019



0.35 ms

Line 182-184 in File LandSaleWithETHAndDAI.sol

```
182     /*@CTK isSANDEnabled
183         @post __return == _sandEnabled
184     */
```

Line 185-187 in File LandSaleWithETHAndDAI.sol

```
185     function isSANDEnabled() external view returns (bool) {
186         return _sandEnabled;
187     }
```

✓ The code meets the specification.

Formal Verification Request 75

If method completes, integer overflow would not happen.



10, Dec 2019



18.17 ms

Line 189 in File LandSaleWithETHAndDAI.sol

```
189     //@CTK NO_OVERFLOW
```

Line 197-221 in File LandSaleWithETHAndDAI.sol

```
197     function _checkValidity(
198         address buyer,
199         address reserved,
200         uint256 x,
201         uint256 y,
202         uint256 size,
203         uint256 price,
204         bytes32 salt,
205         bytes32[] memory proof
206     ) internal view {
207         /* solium-disable-next-line security/no-block-members */
208         require(block.timestamp < _expiryTime, "sale is over");
209         require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
            authorized");
210         require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
```

```

211     bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
212
213     require(
214         _verify(proof, leaf),
215         "Invalid land provided"
216     );
217 }

```

✓ The code meets the specification.

Formal Verification Request 76

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

📅 10, Dec 2019

🕒 0.65 ms

Line 190 in File LandSaleWithETHAndDAI.sol

```
190 // @CTK NO_BUF_OVERFLOW
```

Line 197-221 in File LandSaleWithETHAndDAI.sol

```

197     function _checkValidity(
198         address buyer,
199         address reserved,
200         uint256 x,
201         uint256 y,
202         uint256 size,
203         uint256 price,
204         bytes32 salt,
205         bytes32[] memory proof
206     ) internal view {
207         /* solium-disable-next-line security/no-block-members */
208         require(block.timestamp < _expiryTime, "sale is over");
209         require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
            authorized");
210         require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
211         bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
212
213         require(
214             _verify(proof, leaf),
215             "Invalid land provided"
216         );
217     }

```

✓ The code meets the specification.

Formal Verification Request 77

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.64 ms

Line 191 in File LandSaleWithETHAndDAI.sol

191 //CTK NO_ASF

Line 197-221 in File LandSaleWithETHAndDAI.sol

```

197 function _checkValidity(
198     address buyer,
199     address reserved,
200     uint256 x,
201     uint256 y,
202     uint256 size,
203     uint256 price,
204     bytes32 salt,
205     bytes32[] memory proof
206 ) internal view {
207     /* solium-disable-next-line security/no-block-members */
208     require(block.timestamp < _expiryTime, "sale is over");
209     require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
        authorized");
210     require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
211     bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
212
213     require(
214         _verify(proof, leaf),
215         "Invalid land provided"
216     );
217 }

```

✓ The code meets the specification.

Formal Verification Request 78

_checkValidity



10, Dec 2019



1.51 ms

Line 192-196 in File LandSaleWithETHAndDAI.sol

```

192 /*CTK _checkValidity
193 @tag assume_completion
194 @post buyer == msg.sender /\ _metaTransactionContracts[msg.sender] == true
195 @post reserved == address(0) /\ reserved == buyer
196 */

```

Line 197-221 in File LandSaleWithETHAndDAI.sol

```

197 function _checkValidity(
198     address buyer,
199     address reserved,
200     uint256 x,
201     uint256 y,
202     uint256 size,
203     uint256 price,
204     bytes32 salt,
205     bytes32[] memory proof
206 ) internal view {
207     /* solium-disable-next-line security/no-block-members */
208     require(block.timestamp < _expiryTime, "sale is over");

```

```

209     require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
        authorized");
210     require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
211     bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
212
213     require(
214         _verify(proof, leaf),
215         "Invalid land provided"
216     );
217 }

```

✓ The code meets the specification.

Formal Verification Request 79

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 33.25 ms

Line 243 in File LandSaleWithETHAndDAI.sol

```

243     // @CTK NO_OVERFLOW

```

Line 252-276 in File LandSaleWithETHAndDAI.sol

```

252     function buyLandWithSand(
253         address buyer,
254         address to,
255         address reserved,
256         uint256 x,
257         uint256 y,
258         uint256 size,
259         uint256 priceInSand,
260         bytes32 salt,
261         bytes32[] calldata proof
262     ) external {
263         require(_sandEnabled, "sand payments not enabled");
264         _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
265         require(
266             _sand.transferFrom(
267                 buyer,
268                 _wallet,
269                 priceInSand
270             ),
271             "sand token transfer failed"
272         );
273         _mint(buyer, to, x, y, size, priceInSand, address(_sand), priceInSand);
274     }


```

✓ The code meets the specification.

Formal Verification Request 80

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

📅 10, Dec 2019

 0.54 ms

Line 244 in File LandSaleWithETHAndDAI.sol

244 `//@CTK NO_BUF_OVERFLOW`


Line 252-276 in File LandSaleWithETHAndDAI.sol

```
252     function buyLandWithSand(  
253         address buyer,  
254         address to,  
255         address reserved,  
256         uint256 x,  
257         uint256 y,  
258         uint256 size,  
259         uint256 priceInSand,  
260         bytes32 salt,  
261         bytes32[] calldata proof  
262     ) external {  
263         require(!_sandEnabled, "sand payments not enabled");  
264         _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);  
265         require(  
266             _sand.transferFrom(  
267                 buyer,  
268                 _wallet,  
269                 priceInSand  
270             ),  
271             "sand token transfer failed"  
272         );  
273         _mint(buyer, to, x, y, size, priceInSand, address(_sand), priceInSand);  
274     }
```

 The code meets the specification.

Formal Verification Request 81

Method will not encounter an assertion failure.

 10, Dec 2019 0.53 ms

Line 245 in File LandSaleWithETHAndDAI.sol

245 `//@CTK NO_ASF`

Line 252-276 in File LandSaleWithETHAndDAI.sol

```
252     function buyLandWithSand(  
253         address buyer,  
254         address to,  
255         address reserved,  
256         uint256 x,  
257         uint256 y,  
258         uint256 size,  
259         uint256 priceInSand,  
260         bytes32 salt,  
261         bytes32[] calldata proof  
262     ) external {
```

```

263     require(_sandEnabled, "sand payments not enabled");
264     _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
265     require(
266         _sand.transferFrom(
267             buyer,
268             _wallet,
269             priceInSand
270         ),
271         "sand token transfer failed"
272     );
273     _mint(buyer, to, x, y, size, priceInSand, address(_sand), priceInSand);
274 }

```

✓ The code meets the specification.

Formal Verification Request 82

buyLandWithSand



10, Dec 2019



3.41 ms

Line 246-251 in File LandSaleWithETHAndDAI.sol

```

246     /*@CTK buyLandWithSand
247     @tag assume_completion
248     @post _sandEnabled == true
249     @post buyer == msg.sender \/_ metaTransactionContracts[msg.sender] == true
250     @post reserved == address(0) \/_ reserved == buyer
251     */

```

Line 252-276 in File LandSaleWithETHAndDAI.sol

```

252     function buyLandWithSand(
253         address buyer,
254         address to,
255         address reserved,
256         uint256 x,
257         uint256 y,
258         uint256 size,
259         uint256 priceInSand,
260         bytes32 salt,
261         bytes32[] calldata proof
262     ) external {
263         require(_sandEnabled, "sand payments not enabled");
264         _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
265         require(
266             _sand.transferFrom(
267                 buyer,
268                 _wallet,
269                 priceInSand
270             ),
271             "sand token transfer failed"
272         );
273         _mint(buyer, to, x, y, size, priceInSand, address(_sand), priceInSand);
274     }

```

✓ The code meets the specification.

Formal Verification Request 83

If method completes, integer overflow would not happen.



10, Dec 2019



188.53 ms

Line 290 in File LandSaleWithETHAndDAI.sol

```
290  //@CTK NO_OVERFLOW
```

Line 299-324 in File LandSaleWithETHAndDAI.sol

```

299  function buyLandWithETH(
300      address buyer,
301      address to,
302      address reserved,
303      uint256 x,
304      uint256 y,
305      uint256 size,
306      uint256 priceInSand,
307      bytes32 salt,
308      bytes32[] calldata proof
309  ) external payable {
310      require(_etherEnabled, "ether payments not enabled");
311      _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
312
313      uint256 ETHRequired = getEtherAmountWithSAND(priceInSand);
314      require(msg.value >= ETHRequired, "not enough ether sent");
315      uint256 leftOver = msg.value - ETHRequired;
316      if(leftOver > 0) {
317          msg.sender.transfer(leftOver); // refund extra
318      }
319      address(_wallet).transfer(ETHRequired);
320
321      _mint(buyer, to, x, y, size, priceInSand, address(0), ETHRequired);
322  }

```

✓ The code meets the specification.

Formal Verification Request 84

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



10, Dec 2019



19.28 ms

Line 291 in File LandSaleWithETHAndDAI.sol

```
291  //@CTK NO_BUF_OVERFLOW
```

Line 299-324 in File LandSaleWithETHAndDAI.sol

```

299  function buyLandWithETH(
300      address buyer,
301      address to,
302      address reserved,
303      uint256 x,
304      uint256 y,

```

```

305     uint256 size,
306     uint256 priceInSand,
307     bytes32 salt,
308     bytes32[] calldata proof
309 ) external payable {
310     require(!_etherEnabled, "ether payments not enabled");
311     _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
312
313     uint256 ETHRequired = getEtherAmountWithSAND(priceInSand);
314     require(msg.value >= ETHRequired, "not enough ether sent");
315     uint256 leftOver = msg.value - ETHRequired;
316     if(leftOver > 0) {
317         msg.sender.transfer(leftOver); // refund extra
318     }
319     address(_wallet).transfer(ETHRequired);
320
321     _mint(buyer, to, x, y, size, priceInSand, address(0), ETHRequired);
322 }

```

✓ The code meets the specification.

Formal Verification Request 85

Method will not encounter an assertion failure.



10, Dec 2019



49.79 ms

Line 292 in File LandSaleWithETHAndDAI.sol

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

Line 299-324 in File LandSaleWithETHAndDAI.sol

```

299     function buyLandWithETH(
300         address buyer,
301         address to,
302         address reserved,
303         uint256 x,
304         uint256 y,
305         uint256 size,
306         uint256 priceInSand,
307         bytes32 salt,
308         bytes32[] calldata proof
309     ) external payable {
310         require(!_etherEnabled, "ether payments not enabled");
311         _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
312
313         uint256 ETHRequired = getEtherAmountWithSAND(priceInSand);
314         require(msg.value >= ETHRequired, "not enough ether sent");
315         uint256 leftOver = msg.value - ETHRequired;
316         if(leftOver > 0) {
317             msg.sender.transfer(leftOver); // refund extra
318         }
319         address(_wallet).transfer(ETHRequired);
320
321         _mint(buyer, to, x, y, size, priceInSand, address(0), ETHRequired);
322     }

```


[illegible]

```

54         "key": "ALL_OTHERS",
55         "value": false
56     }
57 ],
58     "_admin": 0
59 }
60 }
61 },
62 {
63     "key": "ALL_OTHERS",
64     "value": "EmptyAddress"
65 }
66 ]
67
68 Function invocation is reverted.

```

Formal Verification Request 86

buyLandWithETH_require



10, Dec 2019



25.67 ms

Line 293-298 in File LandSaleWithETHAndDAI.sol

```

293  /*@CTK buyLandWithETH_require
294     @tag assume_completion
295     @post _sandEnabled == true
296     @post buyer == msg.sender /\ _metaTransactionContracts[msg.sender] == true
297     @post reserved == address(0) /\ reserved == buyer
298  */

```

Line 299-324 in File LandSaleWithETHAndDAI.sol

```

299  function buyLandWithETH(
300      address buyer,
301      address to,
302      address reserved,
303      uint256 x,
304      uint256 y,
305      uint256 size,
306      uint256 priceInSand,
307      bytes32 salt,
308      bytes32[] calldata proof
309  ) external payable {
310      require(_etherEnabled, "ether payments not enabled");
311      _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
312
313      uint256 ETHRequired = getEtherAmountWithSAND(priceInSand);
314      require(msg.value >= ETHRequired, "not enough ether sent");
315      uint256 leftOver = msg.value - ETHRequired;
316      if(leftOver > 0) {
317          msg.sender.transfer(leftOver); // refund extra
318      }
319      address(_wallet).transfer(ETHRequired);
320
321      _mint(buyer, to, x, y, size, priceInSand, address(0), ETHRequired);
322  }

```

✓ The code meets the specification.

Formal Verification Request 87

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 69.69 ms

Line 338 in File LandSaleWithETHAndDAI.sol

338 `//@CTK NO_OVERFLOW`

Line 347-367 in File LandSaleWithETHAndDAI.sol

```
347     function buyLandWithDAI(  
348         address buyer,  
349         address to,  
350         address reserved,  
351         uint256 x,  
352         uint256 y,  
353         uint256 size,  
354         uint256 priceInSand,  
355         bytes32 salt,  
356         bytes32[] calldata proof  
357     ) external {  
358         require(_daiEnabled, "dai payments not enabled");  
359         _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);  
360  
361         uint256 DAIRequired = priceInSand.mul(daiPrice).div(1000000000000000000);  
362         require(_dai.transferFrom(msg.sender, _wallet, DAIRequired), "failed to transfer dai  
363             ");  
364         _mint(buyer, to, x, y, size, priceInSand, address(_dai), DAIRequired);  
365     }
```

✓ The code meets the specification.

Formal Verification Request 88

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

📅 10, Dec 2019

🕒 11.17 ms

Line 339 in File LandSaleWithETHAndDAI.sol

339 `//@CTK NO_BUF_OVERFLOW`

Line 347-367 in File LandSaleWithETHAndDAI.sol

```
347     function buyLandWithDAI(  
348         address buyer,  
349         address to,  
350         address reserved,  
351         uint256 x,  
352         uint256 y,
```

```

353     uint256 size,
354     uint256 priceInSand,
355     bytes32 salt,
356     bytes32[] calldata proof
357 ) external {
358     require(!_daiEnabled, "dai payments not enabled");
359     _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
360
361     uint256 DAIRequired = priceInSand.mul(daiPrice).div(1000000000000000000);
362     require(!_dai.transferFrom(msg.sender, _wallet, DAIRequired), "failed to transfer dai
363         ");
364     _mint(buyer, to, x, y, size, priceInSand, address(_dai), DAIRequired);
365 }


```

✓ The code meets the specification.

Formal Verification Request 89

Method will not encounter an assertion failure.

 10, Dec 2019

 28.57 ms

Line 340 in File LandSaleWithETHAndDAI.sol

```

340    //@CTK FAIL NO_ASF

```

Line 347-367 in File LandSaleWithETHAndDAI.sol

```

347    function buyLandWithDAI(
348        address buyer,
349        address to,
350        address reserved,
351        uint256 x,
352        uint256 y,
353        uint256 size,
354        uint256 priceInSand,
355        bytes32 salt,
356        bytes32[] calldata proof
357    ) external {
358        require(!_daiEnabled, "dai payments not enabled");
359        _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
360
361        uint256 DAIRequired = priceInSand.mul(daiPrice).div(1000000000000000000);
362        require(!_dai.transferFrom(msg.sender, _wallet, DAIRequired), "failed to transfer dai
363            ");
364        _mint(buyer, to, x, y, size, priceInSand, address(_dai), DAIRequired);
365    }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3     Input = {
4         buyer = 0
5         priceInSand = 14

```

```

6      proof = []
7      reserved = 0
8      salt = "RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR"
9      size = 0
10     to = 0
11     x = 0
12     y = 0
13 }
14 This = 0
15 Internal = {
16     __has_assertion_failure = false
17     __has_buf_overflow = false
18     __has_overflow = false
19     __has_returned = false
20     __reverted = false
21     msg = {
22         "gas": 0,
23         "sender": 0,
24         "value": 0
25     }
26 }
27 Other = {
28     block = {
29         "number": 0,
30         "timestamp": 0
31     }
32 }
33 Address_Map = [
34     {
35         "key": "ALL_OTHERS",
36         "value": {
37             "contract_name": "LandSaleWithETHAndDAI",
38             "balance": 0,
39             "contract": {
40                 "GRID_SIZE": 0,
41                 "daiPrice": 17,
42                 "_land": 0,
43                 "_sand": 0,
44                 "_medianizer": 0,
45                 "_dai": 0,
46                 "_wallet": 0,
47                 "_expiryTime": 0,
48                 "_merkleRoot": "RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR",
49                 "_sandEnabled": false,
50                 "_etherEnabled": false,
51                 "_daiEnabled": true,
52                 "_metaTransactionContracts": [
53                     {
54                         "key": "ALL_OTHERS",
55                         "value": false
56                     }
57                 ],
58                 "_admin": 0
59             }
60         }
61     }
62 ]
63

```

64 Function invocation is reverted.

Formal Verification Request 90

buyLandWithDAI



10, Dec 2019



11.77 ms

Line 341-346 in File LandSaleWithETHAndDAI.sol

```
341  /*@CTK buyLandWithDAI
342    @tag assume_completion
343    @post _sandEnabled == true
344    @post buyer == msg.sender /\ _metaTransactionContracts[msg.sender] == true
345    @post reserved == address(0) /\ reserved == buyer
346  */
```

Line 347-367 in File LandSaleWithETHAndDAI.sol

```
347  function buyLandWithDAI(
348    address buyer,
349    address to,
350    address reserved,
351    uint256 x,
352    uint256 y,
353    uint256 size,
354    uint256 priceInSand,
355    bytes32 salt,
356    bytes32[] calldata proof
357  ) external {
358    require(!_daiEnabled, "dai payments not enabled");
359    _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
360
361    uint256 DAIRequired = priceInSand.mul(daiPrice).div(1000000000000000000);
362    require(_dai.transferFrom(msg.sender, _wallet, DAIRequired), "failed to transfer dai");
363
364    _mint(buyer, to, x, y, size, priceInSand, address(_dai), DAIRequired);
365  }
```

✓ The code meets the specification.

Formal Verification Request 91

getExpiryTime



10, Dec 2019



4.0 ms

Line 373-375 in File LandSaleWithETHAndDAI.sol

```
373  /*@CTK getExpiryTime
374    @post __return == _expiryTime
375  */
```

Line 376-378 in File LandSaleWithETHAndDAI.sol

```
376     function getExpiryTime() external view returns(uint256) {
377         return _expiryTime;
378     }
```

✓ The code meets the specification.

Formal Verification Request 92

merkleRoot



10, Dec 2019



4.0 ms

Line 384-386 in File LandSaleWithETHAndDAI.sol

```
384     /*@CTK merkleRoot
385     @post __return == _merkleRoot
386     */
```

Line 387-389 in File LandSaleWithETHAndDAI.sol

```
387     function merkleRoot() external view returns(bytes32) {
388         return _merkleRoot;
389     }
```

✓ The code meets the specification.

Formal Verification Request 93

If method completes, integer overflow would not happen.



10, Dec 2019



6.5 ms

Line 391 in File LandSaleWithETHAndDAI.sol

```
391     //@CTK NO_OVERFLOW
```

Line 394-414 in File LandSaleWithETHAndDAI.sol

```
394     function _generateLandHash(
395         uint256 x,
396         uint256 y,
397         uint256 size,
398         uint256 price,
399         address reserved,
400         bytes32 salt
401     ) internal pure returns (
402         bytes32
403     ) {
404         return keccak256(
405             abi.encodePacked(
406                 x,
407                 y,
408                 size,
409                 price,
410                 reserved,
```

```
411         salt
412     )
413 );
414 }
```

✓ The code meets the specification.

Formal Verification Request 94

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

📅 10, Dec 2019

🕒 0.28 ms

Line 392 in File LandSaleWithETHAndDAI.sol

```
392 // @CTK_NO_BUF_OVERFLOW
```

Line 394-414 in File LandSaleWithETHAndDAI.sol

```
394 function _generateLandHash(
395     uint256 x,
396     uint256 y,
397     uint256 size,
398     uint256 price,
399     address reserved,
400     bytes32 salt
401 ) internal pure returns (
402     bytes32
403 ) {
404     return keccak256(
405         abi.encodePacked(
406             x,
407             y,
408             size,
409             price,
410             reserved,
411             salt
412         )
413     );
414 }
```

✓ The code meets the specification.

Formal Verification Request 95

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.28 ms

Line 393 in File LandSaleWithETHAndDAI.sol

```
393 // @CTK_NO_ASF
```

Line 394-414 in File LandSaleWithETHAndDAI.sol


```
394     function _generateLandHash(  
395         uint256 x,  
396         uint256 y,  
397         uint256 size,  
398         uint256 price,  
399         address reserved,  
400         bytes32 salt  
401     ) internal pure returns (  
402         bytes32  
403     ) {  
404         return keccak256(  
405             abi.encodePacked(  
406                 x,  
407                 y,  
408                 size,  
409                 price,  
410                 reserved,  
411                 salt  
412             )  
413         );  
414     }
```

✓ The code meets the specification.

Formal Verification Request 96

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 0.64 ms

Line 437 in File LandSaleWithETHAndDAI.sol

```
437     // @CTK_NO_OVERFLOW
```

Line 440-443 in File LandSaleWithETHAndDAI.sol

```
440     function getEtherAmountWithSAND(uint256 sandAmount) public view returns (uint256) {  
441         uint256 ethUsdPair = getEthUsdPair();  
442         return sandAmount.mul(daiPrice).div(ethUsdPair);  
443     }
```

✓ The code meets the specification.

Formal Verification Request 97

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

📅 10, Dec 2019

🕒 0.68 ms

Line 438 in File LandSaleWithETHAndDAI.sol

```
438     // @CTK_NO_BUF_OVERFLOW
```

Line 440-443 in File LandSaleWithETHAndDAI.sol

```

440     function getEtherAmountWithSAND(uint256 sandAmount) public view returns (uint256) {
441         uint256 ethUsdPair = getEthUsdPair();
442         return sandAmount.mul(daiPrice).div(ethUsdPair);
443     }

```

✓ The code meets the specification.

Formal Verification Request 98

Method will not encounter an assertion failure.



10, Dec 2019



8.03 ms

Line 439 in File LandSaleWithETHAndDAI.sol

```

439     //@CTK FAIL NO_ASF

```

Line 440-443 in File LandSaleWithETHAndDAI.sol

```

440     function getEtherAmountWithSAND(uint256 sandAmount) public view returns (uint256) {
441         uint256 ethUsdPair = getEthUsdPair();
442         return sandAmount.mul(daiPrice).div(ethUsdPair);
443     }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3     Input = {
4         sandAmount = 59
5     }
6     This = 0
7     Internal = {
8         __has_assertion_failure = false
9         __has_buf_overflow = false
10        __has_overflow = false
11        __has_returned = false
12        __reverted = false
13        msg = {
14            "gas": 0,
15            "sender": 0,
16            "value": 0
17        }
18    }
19    Other = {
20        __return = 0
21        block = {
22            "number": 0,
23            "timestamp": 0
24        }
25    }
26    Address_Map = [
27        {
28            "key": "ALL_OTHERS",
29            "value": {
30                "contract_name": "LandSaleWithETHAndDAI",
31                "balance": 0,

```

```


32     "contract": {
33         "GRID_SIZE": 0,
34         "daiPrice": 0,
35         "_land": 0,
36         "_sand": 0,
37         "_medianizer": 0,
38         "_dai": 0,
39         "_wallet": 0,
40         "_expiryTime": 0,
41         "_merkleRoot": "|||||",
42         "_sandEnabled": false,
43         "_etherEnabled": false,
44         "_daiEnabled": false,
45         "_metaTransactionContracts": [
46             {
47                 "key": "ALL_OTHERS",
48                 "value": false
49             }
50         ],
51         "_admin": 0
52     }
53 }
54 }
55 ]
56
57 Function invocation is reverted.

```

Formal Verification Request 99

getAdmin

 10, Dec 2019

 4.67 ms

Line 11-13 in File Admin.sol

```

11     /*@CTK getAdmin
12     @post __return == _admin
13     */

```

Line 14-16 in File Admin.sol

```

14     function getAdmin() external view returns (address) {
15         return _admin;
16     }


```

 The code meets the specification.

Formal Verification Request 100

If method completes, integer overflow would not happen.

 10, Dec 2019

 13.15 ms

Line 20 in File Admin.sol

20 `//@CTK NO_OVERFLOW`

Line 32-36 in File Admin.sol


```
32 function changeAdmin(address newAdmin) external {
33     require(msg.sender == _admin, "only admin can change admin");
34     emit AdminChanged(_admin, newAdmin);
35     _admin = newAdmin;
36 }
```

✓ The code meets the specification.

Formal Verification Request 101

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

 10, Dec 2019

 0.38 ms

Line 21 in File Admin.sol

21 `//@CTK NO_BUF_OVERFLOW`

Line 32-36 in File Admin.sol


```
32 function changeAdmin(address newAdmin) external {
33     require(msg.sender == _admin, "only admin can change admin");
34     emit AdminChanged(_admin, newAdmin);
35     _admin = newAdmin;
36 }
```

✓ The code meets the specification.

Formal Verification Request 102

Method will not encounter an assertion failure.

 10, Dec 2019

 0.38 ms

Line 22 in File Admin.sol

22 `//@CTK NO_ASF`

Line 32-36 in File Admin.sol

```
32 function changeAdmin(address newAdmin) external {
33     require(msg.sender == _admin, "only admin can change admin");
34     emit AdminChanged(_admin, newAdmin);
35     _admin = newAdmin;
36 }
```

✓ The code meets the specification.

Formal Verification Request 103

changeAdmin_requirement



10, Dec 2019



0.77 ms

Line 23-26 in File Admin.sol

```
23  /*@CTK changeAdmin_requirement
24    @tag assume_completion
25    @post msg.sender == _admin
26  */
```

Line 32-36 in File Admin.sol

```
32  function changeAdmin(address newAdmin) external {
33    require(msg.sender == _admin, "only admin can change admin");
34    emit AdminChanged(_admin, newAdmin);
35    _admin = newAdmin;
36  }
```

✓ The code meets the specification.

Formal Verification Request 104

changeAdmin_change



10, Dec 2019



1.13 ms

Line 27-31 in File Admin.sol

```
27  /*@CTK changeAdmin_change
28    @tag assume_completion
29    @pre msg.sender == _admin
30    @post __post._admin == newAdmin
31  */
```

Line 32-36 in File Admin.sol

```
32  function changeAdmin(address newAdmin) external {
33    require(msg.sender == _admin, "only admin can change admin");
34    emit AdminChanged(_admin, newAdmin);
35    _admin = newAdmin;
36  }
```

✓ The code meets the specification.

Formal Verification Request 105

If method completes, integer overflow would not happen.



10, Dec 2019



25.15 ms

Line 13 in File MetaTransactionReceiver.sol

13 `//@CTK NO_OVERFLOW`

Line 25-31 in File MetaTransactionReceiver.sol

```
25     function setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
      public {
26         require(
27             msg.sender == _admin,
28             "only admin can setup metaTransactionProcessors"
29         );
30         _setMetaTransactionProcessor(metaTransactionProcessor, enabled);
31     }
```

✓ The code meets the specification.

Formal Verification Request 106

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

📅 10, Dec 2019

🕒 0.41 ms

Line 14 in File MetaTransactionReceiver.sol

14 `//@CTK NO_BUF_OVERFLOW`

Line 25-31 in File MetaTransactionReceiver.sol

```
25     function setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
      public {
26         require(
27             msg.sender == _admin,
28             "only admin can setup metaTransactionProcessors"
29         );
30         _setMetaTransactionProcessor(metaTransactionProcessor, enabled);
31     }
```

✓ The code meets the specification.

Formal Verification Request 107

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.41 ms

Line 15 in File MetaTransactionReceiver.sol

15 `//@CTK NO_ASF`

Line 25-31 in File MetaTransactionReceiver.sol

```
25     function setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
      public {
26         require(
27             msg.sender == _admin,
28             "only admin can setup metaTransactionProcessors"
29         );
```

```
30     _setMetaTransactionProcessor(metaTransactionProcessor, enabled);
31 }
```

✓ The code meets the specification.

Formal Verification Request 108

setMetaTransactionProcessor



10, Dec 2019



1.04 ms

Line 16-19 in File MetaTransactionReceiver.sol

```
16  /*@CTK setMetaTransactionProcessor
17  @tag assume_completion
18  @post msg.sender == _admin
19  */
```

Line 25-31 in File MetaTransactionReceiver.sol

```
25  function setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
26  public {
27  require(
28      msg.sender == _admin,
29      "only admin can setup metaTransactionProcessors"
30  );
31  _setMetaTransactionProcessor(metaTransactionProcessor, enabled);
32  }
```

✓ The code meets the specification.

Formal Verification Request 109

setMetaTransactionProcessor



10, Dec 2019



1.04 ms

Line 20-24 in File MetaTransactionReceiver.sol

```
20  /*@CTK setMetaTransactionProcessor
21  @tag assume_completion
22  @inv msg.sender == _admin
23  @post __post._metaTransactionContracts[metaTransactionProcessor] == enabled
24  */
```

Line 25-31 in File MetaTransactionReceiver.sol

```
25  function setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
26  public {
27  require(
28      msg.sender == _admin,
29      "only admin can setup metaTransactionProcessors"
30  );
31  _setMetaTransactionProcessor(metaTransactionProcessor, enabled);
32  }
```

✓ The code meets the specification.

Formal Verification Request 110

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 0.33 ms

Line 33 in File MetaTransactionReceiver.sol

```
33 // @CTK_NO_OVERFLOW
```

Line 40-43 in File MetaTransactionReceiver.sol

```
40 function _setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
    internal {
41     _metaTransactionContracts[metaTransactionProcessor] = enabled;
42     emit MetaTransactionProcessor(metaTransactionProcessor, enabled);
43 }
```

✓ The code meets the specification.

Formal Verification Request 111

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

📅 10, Dec 2019

🕒 0.26 ms

Line 34 in File MetaTransactionReceiver.sol

```
34 // @CTK_NO_BUF_OVERFLOW
```

Line 40-43 in File MetaTransactionReceiver.sol

```
40 function _setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
    internal {
41     _metaTransactionContracts[metaTransactionProcessor] = enabled;
42     emit MetaTransactionProcessor(metaTransactionProcessor, enabled);
43 }
```

✓ The code meets the specification.

Formal Verification Request 112

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.26 ms

Line 35 in File MetaTransactionReceiver.sol

```
35 // @CTK_NO_ASF
```

Line 40-43 in File MetaTransactionReceiver.sol


```

40  function _setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
      internal {
41      _metaTransactionContracts[metaTransactionProcessor] = enabled;
42      emit MetaTransactionProcessor(metaTransactionProcessor, enabled);
43  }


```

✓ The code meets the specification.

Formal Verification Request 113

`__setMetaTransactionProcessor`

 10, Dec 2019

 1.01 ms

Line 36-39 in File MetaTransactionReceiver.sol

```

36  /*@CTK __setMetaTransactionProcessor
37  @tag assume_completion
38  @post __post._metaTransactionContracts[metaTransactionProcessor] == enabled
39  */

```

Line 40-43 in File MetaTransactionReceiver.sol

```

40  function _setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
      internal {
41      _metaTransactionContracts[metaTransactionProcessor] = enabled;
42      emit MetaTransactionProcessor(metaTransactionProcessor, enabled);
43  }


```

✓ The code meets the specification.

Formal Verification Request 114

If method completes, integer overflow would not happen.

 10, Dec 2019

 4.25 ms

Line 48 in File MetaTransactionReceiver.sol

```

48  //@CTK NO_OVERFLOW

```

Line 55-57 in File MetaTransactionReceiver.sol

```

55  function isMetaTransactionProcessor(address who) external view returns(bool) {
56      return _metaTransactionContracts[who];
57  }

```

✓ The code meets the specification.

Formal Verification Request 115

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

📅 10, Dec 2019

🕒 0.5 ms

Line 49 in File MetaTransactionReceiver.sol

```
49  // @CTK NO_BUF_OVERFLOW
```

Line 55-57 in File MetaTransactionReceiver.sol

```
55  function isMetaTransactionProcessor(address who) external view returns(bool) {  
56      return _metaTransactionContracts[who];  
57  }
```

✅ The code meets the specification.

Formal Verification Request 116

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.74 ms

Line 50 in File MetaTransactionReceiver.sol

```
50  // @CTK NO_ASF
```

Line 55-57 in File MetaTransactionReceiver.sol

```
55  function isMetaTransactionProcessor(address who) external view returns(bool) {  
56      return _metaTransactionContracts[who];  
57  }
```

✅ The code meets the specification.

Formal Verification Request 117

isMetaTransactionProcessor

📅 10, Dec 2019

🕒 0.38 ms

Line 51-54 in File MetaTransactionReceiver.sol

```
51  /* @CTK isMetaTransactionProcessor  
52      @tag assume_completion  
53      @post __return == _metaTransactionContracts[who]  
54  */
```

Line 55-57 in File MetaTransactionReceiver.sol

```
55  function isMetaTransactionProcessor(address who) external view returns(bool) {  
56      return _metaTransactionContracts[who];  
57  }
```

✅ The code meets the specification.

Formal Verification Request 118

If method completes, integer overflow would not happen.



10, Dec 2019



24.14 ms

Line 25 in File ERC721BaseToken.sol

```
25 // @CTK_NO_OVERFLOW
```

Line 33-39 in File ERC721BaseToken.sol

```
33 constructor(  
34     address metaTransactionContract,  
35     address admin  
36 ) internal {  
37     _admin = admin;  
38     _setMetaTransactionProcessor(metaTransactionContract, true);  
39 }
```

✓ The code meets the specification.

Formal Verification Request 119

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



10, Dec 2019



0.47 ms

Line 26 in File ERC721BaseToken.sol

```
26 // @CTK_NO_BUF_OVERFLOW
```

Line 33-39 in File ERC721BaseToken.sol

```
33 constructor(  
34     address metaTransactionContract,  
35     address admin  
36 ) internal {  
37     _admin = admin;  
38     _setMetaTransactionProcessor(metaTransactionContract, true);  
39 }
```

✓ The code meets the specification.

Formal Verification Request 120

Method will not encounter an assertion failure.



10, Dec 2019



0.43 ms

Line 27 in File ERC721BaseToken.sol

```
27 // @CTK_NO_ASF
```

Line 33-39 in File ERC721BaseToken.sol

```

33     constructor(
34         address metaTransactionContract,
35         address admin
36     ) internal {
37         _admin = admin;
38         _setMetaTransactionProcessor(metaTransactionContract, true);
39     }

```

✓ The code meets the specification.

Formal Verification Request 121

ERC721BaseToken

📅 10, Dec 2019

🕒 1.57 ms

Line 28-32 in File ERC721BaseToken.sol

```

28     /*@CTK ERC721BaseToken
29         @tag assume_completion
30         @post __post._admin == admin
31         @post __post._metaTransactionContracts[metaTransactionContract] == true
32     */

```

Line 33-39 in File ERC721BaseToken.sol

```

33     constructor(
34         address metaTransactionContract,
35         address admin
36     ) internal {
37         _admin = admin;
38         _setMetaTransactionProcessor(metaTransactionContract, true);
39     }

```

✓ The code meets the specification.

Formal Verification Request 122

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 86.6 ms

Line 41 in File ERC721BaseToken.sol

```

41     //@CTK FAIL NO_OVERFLOW

```

Line 53-58 in File ERC721BaseToken.sol

```

53     function _transferFrom(address from, address to, uint256 id) internal {
54         _numNFTPerAddress[from]--;
55         _numNFTPerAddress[to]++;
56         _owners[id] = uint256(to);
57         emit Transfer(from, to, id);
58     }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     from = 1
5     id = 0
6     to = 0
7   }
8   This = 0
9   Internal = {
10    __has_assertion_failure = false
11    __has_buf_overflow = false
12    __has_overflow = false
13    __has_returned = false
14    __reverted = false
15    msg = {
16      "gas": 0,
17      "sender": 0,
18      "value": 0
19    }
20  }
21  Other = {
22    block = {
23      "number": 0,
24      "timestamp": 0
25    }
26  }
27  Address_Map = [
28    {
29      "key": 0,
30      "value": {
31        "contract_name": "ERC721BaseToken",
32        "balance": 0,
33        "contract": {
34          "_ERC721_RECEIVED": "AAAA",
35          "_ERC721_BATCH_RECEIVED": "AAAA",
36          "ERC165ID": "AAAA",
37          "ERC721_MANDATORY_RECEIVER": "AAAA",
38          "_numNFTPerAddress": [
39            {
40              "key": 18,
41              "value": 0
42            },
43            {
44              "key": 48,
45              "value": 2
46            },
47            {
48              "key": 192,
49              "value": 0
50            },
51            {
52              "key": 40,
53              "value": 16
54            },
55            {
56              "key": 66,
57              "value": 4
58            },

```

```

59     {
60         "key": 128,
61         "value": 0
62     },
63     {
64         "key": 65,
65         "value": 1
66     },
67     {
68         "key": 1,
69         "value": 0
70     },
71     {
72         "key": 8,
73         "value": 16
74     },
75     {
76         "key": 0,
77         "value": 182
78     },
79     {
80         "key": 16,
81         "value": 0
82     },
83     {
84         "key": 4,
85         "value": 0
86     },
87     {
88         "key": 32,
89         "value": 2
90     },
91     {
92         "key": 2,
93         "value": 2
94     },
95     {
96         "key": 64,
97         "value": 0
98     },
99     {
100         "key": "ALL_OTHERS",
101         "value": 255
102     }
103 ],
104 "_owners": [
105     {
106         "key": 64,
107         "value": 64
108     },
109     {
110         "key": 128,
111         "value": 8
112     },
113     {
114         "key": 1,
115         "value": 1
116     },

```

```

117     {
118         "key": 8,
119         "value": 4
120     },
121     {
122         "key": 0,
123         "value": 2
124     },
125     {
126         "key": 16,
127         "value": 1
128     },
129     {
130         "key": 4,
131         "value": 4
132     },
133     {
134         "key": 32,
135         "value": 8
136     },
137     {
138         "key": "ALL_OTHERS",
139         "value": 0
140     }
141 ],
142 "_operatorsForAll": [
143     {
144         "key": "ALL_OTHERS",
145         "value": [
146             {
147                 "key": "ALL_OTHERS",
148                 "value": false
149             }
150         ]
151     }
152 ],
153 "_operators": [
154     {
155         "key": 64,
156         "value": 16
157     },
158     {
159         "key": 128,
160         "value": 32
161     },
162     {
163         "key": 0,
164         "value": 160
165     },
166     {
167         "key": 16,
168         "value": 32
169     },
170     {
171         "key": 2,
172         "value": 128
173     },
174     {

```

```

175         "key": "ALL_OTHERS",
176         "value": 0
177     }
178 ],
179     "_metaTransactionContracts": [
180     {
181         "key": "ALL_OTHERS",
182         "value": false
183     }
184 ],
185     "_admin": 0,
186     "_superOperators": [
187     {
188         "key": 0,
189         "value": true
190     },
191     {
192         "key": "ALL_OTHERS",
193         "value": false
194     }
195 ]
196 }
197 }
198 },
199 {
200     "key": "ALL_OTHERS",
201     "value": "EmptyAddress"
202 }
203 ]
204

```

205 After Execution:

```

206     Input = {
207         from = 1
208         id = 0
209         to = 0
210     }
211     This = 0
212     Internal = {
213         __has_assertion_failure = false
214         __has_buf_overflow = false
215         __has_overflow = true
216         __has_returned = false
217         __reverted = false
218         msg = {
219             "gas": 0,
220             "sender": 0,
221             "value": 0
222         }
223     }
224     Other = {
225         block = {
226             "number": 0,
227             "timestamp": 0
228         }
229     }
230     Address_Map = [
231     {
232         "key": 0,

```



```

233 "value": {
234   "contract_name": "ERC721BaseToken",
235   "balance": 0,
236   "contract": {
237     "_ERC721_RECEIVED": "AAAA",
238     "_ERC721_BATCH_RECEIVED": "AAAA",
239     "ERC165ID": "AAAA",
240     "ERC721_MANDATORY_RECEIVER": "AAAA",
241     "_numNFTPerAddress": [
242       {
243         "key": 18,
244         "value": 0
245       },
246       {
247         "key": 64,
248         "value": 0
249       },
250       {
251         "key": 192,
252         "value": 0
253       },
254       {
255         "key": 40,
256         "value": 16
257       },
258       {
259         "key": 66,
260         "value": 4
261       },
262       {
263         "key": 128,
264         "value": 0
265       },
266       {
267         "key": 65,
268         "value": 1
269       },
270       {
271         "key": 8,
272         "value": 16
273       },
274       {
275         "key": 0,
276         "value": 183
277       },
278       {
279         "key": 16,
280         "value": 0
281       },
282       {
283         "key": 4,
284         "value": 0
285       },
286       {
287         "key": 32,
288         "value": 2
289       },
290       {

```

```

291         "key": 2,
292         "value": 2
293     },
294     {
295         "key": 48,
296         "value": 2
297     },
298     {
299         "key": "ALL_OTHERS",
300         "value": 255
301     }
302 ],
303 "_owners": [
304     {
305         "key": 64,
306         "value": 64
307     },
308     {
309         "key": 128,
310         "value": 8
311     },
312     {
313         "key": 1,
314         "value": 1
315     },
316     {
317         "key": 8,
318         "value": 4
319     },
320     {
321         "key": 16,
322         "value": 1
323     },
324     {
325         "key": 4,
326         "value": 4
327     },
328     {
329         "key": 32,
330         "value": 8
331     },
332     {
333         "key": "ALL_OTHERS",
334         "value": 0
335     }
336 ],
337 "_operatorsForAll": [
338     {
339         "key": "ALL_OTHERS",
340         "value": [
341             {
342                 "key": "ALL_OTHERS",
343                 "value": false
344             }
345         ]
346     }
347 ],
348 "_operators": [

```

```

349     {
350         "key": 64,
351         "value": 16
352     },
353     {
354         "key": 128,
355         "value": 32
356     },
357     {
358         "key": 0,
359         "value": 160
360     },
361     {
362         "key": 16,
363         "value": 32
364     },
365     {
366         "key": 2,
367         "value": 128
368     },
369     {
370         "key": "ALL_OTHERS",
371         "value": 0
372     }
373 ],
374 "_metaTransactionContracts": [
375     {
376         "key": "ALL_OTHERS",
377         "value": false
378     }
379 ],
380 "_admin": 0,
381 "_superOperators": [
382     {
383         "key": 0,
384         "value": true
385     },
386     {
387         "key": "ALL_OTHERS",
388         "value": false
389     }
390 ]
391 }
392 }
393 },
394 {
395     "key": "ALL_OTHERS",
396     "value": "EmptyAddress"
397 }
398 ]

```

Formal Verification Request 123

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



10, Dec 2019



0.39 ms

Line 42 in File ERC721BaseToken.sol

```
42  // @CTK NO_BUF_OVERFLOW
```

Line 53-58 in File ERC721BaseToken.sol


```
53  function _transferFrom(address from, address to, uint256 id) internal {
54      _numNFTPerAddress[from]--;
55      _numNFTPerAddress[to]++;
56      _owners[id] = uint256(to);
57      emit Transfer(from, to, id);
58  }
```

✓ The code meets the specification.

Formal Verification Request 124

Method will not encounter an assertion failure.

 10, Dec 2019

 0.34 ms

Line 43 in File ERC721BaseToken.sol

```
43  // @CTK NO_ASF
```

Line 53-58 in File ERC721BaseToken.sol


```
53  function _transferFrom(address from, address to, uint256 id) internal {
54      _numNFTPerAddress[from]--;
55      _numNFTPerAddress[to]++;
56      _owners[id] = uint256(to);
57      emit Transfer(from, to, id);
58  }
```

✓ The code meets the specification.

Formal Verification Request 125

__transferFrom

 10, Dec 2019

 4.59 ms

Line 44-52 in File ERC721BaseToken.sol

```
44  /* @CTK __transferFrom
45     @tag assume_completion
46     @pre from != to
47     @pre _numNFTPerAddress[from] > 0
48     @pre address(_owners[id]) == from
49     @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
50     @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
51     @post __post._owners[id] == uint256(to)
52  */
```

Line 53-58 in File ERC721BaseToken.sol

```
53     function _transferFrom(address from, address to, uint256 id) internal {  
54         _numNFTPerAddress[from]--;  
55         _numNFTPerAddress[to]++;  
56         _owners[id] = uint256(to);  
57         emit Transfer(from, to, id);  
58     }
```

✓ The code meets the specification.

Formal Verification Request 126

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 11.59 ms

Line 65 in File ERC721BaseToken.sol

```
65     //@CTK_NO_OVERFLOW
```

Line 77-80 in File ERC721BaseToken.sol

```
77     function balanceOf(address owner) external view returns (uint256) {  
78         require(owner != address(0), "owner is zero address");  
79         return _numNFTPerAddress[owner];  
80     }
```

✓ The code meets the specification.

Formal Verification Request 127

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

📅 10, Dec 2019

🕒 0.33 ms

Line 66 in File ERC721BaseToken.sol

```
66     //@CTK_NO_BUF_OVERFLOW
```

Line 77-80 in File ERC721BaseToken.sol

```
77     function balanceOf(address owner) external view returns (uint256) {  
78         require(owner != address(0), "owner is zero address");  
79         return _numNFTPerAddress[owner];  
80     }
```

✓ The code meets the specification.

Formal Verification Request 128

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.32 ms

Line 67 in File ERC721BaseToken.sol

```
67 //CTK NO_ASF
```

Line 77-80 in File ERC721BaseToken.sol

```
77 function balanceOf(address owner) external view returns (uint256) {  
78     require(owner != address(0), "owner is zero address");  
79     return _numNFTPerAddress[owner];  
80 }
```

✓ The code meets the specification.

Formal Verification Request 129

balanceOf__require

📅 10, Dec 2019

🕒 0.32 ms

Line 68-71 in File ERC721BaseToken.sol

```
68 /*CTK balanceOf__require  
69     @tag assume_completion  
70     @post owner != address(0)  
71 */
```

Line 77-80 in File ERC721BaseToken.sol

```
77 function balanceOf(address owner) external view returns (uint256) {  
78     require(owner != address(0), "owner is zero address");  
79     return _numNFTPerAddress[owner];  
80 }
```

✓ The code meets the specification.

Formal Verification Request 130

balanceOf__change

📅 10, Dec 2019

🕒 1.05 ms

Line 72-76 in File ERC721BaseToken.sol

```
72 /*CTK balanceOf__change  
73     @tag assume_completion  
74     @pre owner != address(0)  
75     @post __return == _numNFTPerAddress[owner]  
76 */
```

Line 77-80 in File ERC721BaseToken.sol


```
77 function balanceOf(address owner) external view returns (uint256) {  
78     require(owner != address(0), "owner is zero address");  
79     return _numNFTPerAddress[owner];  
80 }
```

✓ The code meets the specification.

Formal Verification Request 131

If method completes, integer overflow would not happen.

 10, Dec 2019

 4.19 ms

Line 82 in File ERC721BaseToken.sol

```
82 // @CTK_NO_OVERFLOW
```

Line 87-89 in File ERC721BaseToken.sol


```
87 function _ownerOf(uint256 id) internal view returns (address) {  
88     return address(_owners[id]);  
89 }
```

 The code meets the specification.

Formal Verification Request 132

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

 10, Dec 2019

 0.26 ms

Line 83 in File ERC721BaseToken.sol

```
83 // @CTK_NO_BUF_OVERFLOW
```

Line 87-89 in File ERC721BaseToken.sol


```
87 function _ownerOf(uint256 id) internal view returns (address) {  
88     return address(_owners[id]);  
89 }
```

 The code meets the specification.

Formal Verification Request 133

__ownerOf

 10, Dec 2019

 0.27 ms

Line 84-86 in File ERC721BaseToken.sol

```
84 /* @CTK __ownerOf  
85     @post __return == address(_owners[id])  
86 */
```

Line 87-89 in File ERC721BaseToken.sol

```
87 function _ownerOf(uint256 id) internal view returns (address) {  
88     return address(_owners[id]);  
89 }
```

 The code meets the specification.

Formal Verification Request 134

If method completes, integer overflow would not happen.



10, Dec 2019



5.97 ms

Line 91 in File ERC721BaseToken.sol

```
91  // @CTK_NO_OVERFLOW
```

Line 98-102 in File ERC721BaseToken.sol

```
98  function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner,  
99      bool operatorEnabled) {  
100      uint256 data = _owners[id];  
101      owner = address(data);  
102      operatorEnabled = (data / 2**255) == 1;  
103  }
```



The code meets the specification.

Formal Verification Request 135

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



10, Dec 2019



0.27 ms

Line 92 in File ERC721BaseToken.sol

```
92  // @CTK_NO_BUF_OVERFLOW
```

Line 98-102 in File ERC721BaseToken.sol

```
98  function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner,  
99      bool operatorEnabled) {  
100      uint256 data = _owners[id];  
101      owner = address(data);  
102      operatorEnabled = (data / 2**255) == 1;  
103  }
```



The code meets the specification.

Formal Verification Request 136

Method will not encounter an assertion failure.



10, Dec 2019



11.66 ms

Line 93 in File ERC721BaseToken.sol

```
93  // @CTK_FAIL_NO_ASF
```

Line 98-102 in File ERC721BaseToken.sol


```

98  function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner,
    bool operatorEnabled) {
99      uint256 data = _owners[id];
100     owner = address(data);
101     operatorEnabled = (data / 2**255) == 1;
102 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     id = 0
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    block = {
21      "number": 0,
22      "timestamp": 0
23    }
24    operatorEnabled = false
25    owner = 0
26  }
27  Address_Map = [
28    {
29      "key": "ALL_OTHERS",
30      "value": {
31        "contract_name": "ERC721BaseToken",
32        "balance": 0,
33        "contract": {
34          "_ERC721_RECEIVED": "AAAA",
35          "_ERC721_BATCH_RECEIVED": "AAAA",
36          "ERC165ID": "AAAA",
37          "ERC721_MANDATORY_RECEIVER": "CCCC",
38          "_numNFTPerAddress": [
39            {
40              "key": "ALL_OTHERS",
41              "value": 0
42            }
43          ],
44          "_owners": [
45            {
46              "key": 0,
47              "value": 32
48            },
49            {
50              "key": 2,

```

```

51         "value": 34
52     },
53     {
54         "key": 64,
55         "value": 2
56     },
57     {
58         "key": 128,
59         "value": 16
60     },
61     {
62         "key": 16,
63         "value": 0
64     },
65     {
66         "key": "ALL_OTHERS",
67         "value": 8
68     }
69 ],
70 "_operatorsForAll": [
71     {
72         "key": "ALL_OTHERS",
73         "value": [
74             {
75                 "key": "ALL_OTHERS",
76                 "value": true
77             }
78         ]
79     }
80 ],
81 "_operators": [
82     {
83         "key": 2,
84         "value": 16
85     },
86     {
87         "key": 128,
88         "value": 128
89     },
90     {
91         "key": 8,
92         "value": 64
93     },
94     {
95         "key": 64,
96         "value": 4
97     },
98     {
99         "key": "ALL_OTHERS",
100        "value": 0
101    }
102 ],
103 "_metaTransactionContracts": [
104     {
105         "key": "ALL_OTHERS",
106         "value": true
107     }
108 ],

```

```

109         "_admin": 0,
110         "_superOperators": [
111             {
112                 "key": "ALL_OTHERS",
113                 "value": false
114             }
115         ]
116     }
117 }
118 }
119 ]
120
121 Function invocation is reverted.

```

Formal Verification Request 137

`_ownerAndOperatorEnabledOf`



10, Dec 2019



0.44 ms

Line 94-97 in File ERC721BaseToken.sol

```

94     /*@CTK _ownerAndOperatorEnabledOf
95     @post owner == address(_owners[id])
96     @post operatorEnabled == ((_owners[id] / 2**255) == 1)
97     */

```

Line 98-102 in File ERC721BaseToken.sol

```

98     function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner,
99         bool operatorEnabled) {
100         uint256 data = _owners[id];
101         owner = address(data);
102         operatorEnabled = (data / 2**255) == 1;
103     }

```

✓ The code meets the specification.

Formal Verification Request 138

If method completes, integer overflow would not happen.



10, Dec 2019



20.43 ms

Line 109 in File ERC721BaseToken.sol

```

109     //@CTK NO_OVERFLOW

```

Line 117-120 in File ERC721BaseToken.sol

```

117     function ownerOf(uint256 id) external view returns (address owner) {
118         owner = _ownerOf(id);
119         require(owner != address(0), "token does not exist");
120     }

```

✓ The code meets the specification.

Formal Verification Request 139

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



10, Dec 2019



0.39 ms

Line 110 in File ERC721BaseToken.sol

```
110 // @CTK_NO_BUF_OVERFLOW
```

Line 117-120 in File ERC721BaseToken.sol

```
117 function ownerOf(uint256 id) external view returns (address owner) {  
118     owner = _ownerOf(id);  
119     require(owner != address(0), "token does not exist");  
120 }
```



The code meets the specification.

Formal Verification Request 140

Method will not encounter an assertion failure.



10, Dec 2019



0.38 ms

Line 111 in File ERC721BaseToken.sol

```
111 // @CTK_NO_ASF
```

Line 117-120 in File ERC721BaseToken.sol

```
117 function ownerOf(uint256 id) external view returns (address owner) {  
118     owner = _ownerOf(id);  
119     require(owner != address(0), "token does not exist");  
120 }
```



The code meets the specification.

Formal Verification Request 141

ownerOf



10, Dec 2019



0.74 ms

Line 112-116 in File ERC721BaseToken.sol

```
112 /* @CTK ownerOf  
113     @tag assume_completion  
114     @post owner == address(_owners[id])  
115     @post owner != address(0)  
116 */
```

Line 117-120 in File ERC721BaseToken.sol

```

117 function ownerOf(uint256 id) external view returns (address owner) {
118     owner = _ownerOf(id);
119     require(owner != address(0), "token does not exist");
120 }

```

✓ The code meets the specification.

Formal Verification Request 142

If method completes, integer overflow would not happen.



10, Dec 2019



16.47 ms

Line 122 in File ERC721BaseToken.sol

```

122 //CTK NO_OVERFLOW

```

Line 130-138 in File ERC721BaseToken.sol

```

130 function _approveFor(address owner, address operator, uint256 id) internal {
131     if(operator == address(0)) {
132         _owners[id] = uint256(owner); // no need to resset the operator, it will be
            overridden next time
133     } else {
134         _owners[id] = uint256(owner) + 2**255;
135         _operators[id] = operator;
136     }
137     emit Approval(owner, operator, id);
138 }

```

✓ The code meets the specification.

Formal Verification Request 143

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



10, Dec 2019



0.33 ms

Line 123 in File ERC721BaseToken.sol

```

123 //CTK NO_BUF_OVERFLOW

```

Line 130-138 in File ERC721BaseToken.sol

```

130 function _approveFor(address owner, address operator, uint256 id) internal {
131     if(operator == address(0)) {
132         _owners[id] = uint256(owner); // no need to resset the operator, it will be
            overridden next time
133     } else {
134         _owners[id] = uint256(owner) + 2**255;
135         _operators[id] = operator;
136     }
137     emit Approval(owner, operator, id);
138 }

```

✓ The code meets the specification.

Formal Verification Request 144

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.32 ms

Line 124 in File ERC721BaseToken.sol

124 `//@CTK NO_ASF`

Line 130-138 in File ERC721BaseToken.sol

```
130 function _approveFor(address owner, address operator, uint256 id) internal {
131     if(operator == address(0)) {
132         _owners[id] = uint256(owner); // no need to resset the operator, it will be
            overridden next time
133     } else {
134         _owners[id] = uint256(owner) + 2**255;
135         _operators[id] = operator;
136     }
137     emit Approval(owner, operator, id);
138 }
```

✅ The code meets the specification.

Formal Verification Request 145

`__approveFor`

📅 10, Dec 2019

🕒 1.97 ms

Line 125-129 in File ERC721BaseToken.sol

```
125 /*@CTK __approveFor
126     @post (operator == address(0)) -> (__post._owners[id] == uint256(owner))
127     @post (operator != address(0)) -> (__post._owners[id] == uint256(owner) + 2**255)
128     @post (operator != address(0)) -> (__post._operators[id] == operator)
129 */
```

Line 130-138 in File ERC721BaseToken.sol

```
130 function _approveFor(address owner, address operator, uint256 id) internal {
131     if(operator == address(0)) {
132         _owners[id] = uint256(owner); // no need to resset the operator, it will be
            overridden next time
133     } else {
134         _owners[id] = uint256(owner) + 2**255;
135         _operators[id] = operator;
136     }
137     emit Approval(owner, operator, id);
138 }
```

✅ The code meets the specification.

Formal Verification Request 146

If method completes, integer overflow would not happen.



10, Dec 2019



60.82 ms

Line 146 in File ERC721BaseToken.sol

146 `//@CTK NO_OVERFLOW`

Line 164-180 in File ERC721BaseToken.sol

```
164     function approveFor(  
165         address sender,  
166         address operator,  
167         uint256 id  
168     ) external {  
169         address owner = _ownerOf(id);  
170         require(sender != address(0), "sender is zero address");  
171         require(  
172             msg.sender == sender ||  
173             _metaTransactionContracts[msg.sender] ||  
174             _superOperators[msg.sender] ||  
175             _operatorsForAll[sender][msg.sender],  
176             "not authorized to approve"  
177         );  
178         require(owner == sender, "owner != sender");  
179         _approveFor(owner, operator, id);  
180     }
```

✓ The code meets the specification.

Formal Verification Request 147

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



10, Dec 2019



4.85 ms

Line 147 in File ERC721BaseToken.sol

147 `//@CTK NO_BUF_OVERFLOW`

Line 164-180 in File ERC721BaseToken.sol

```
164     function approveFor(  
165         address sender,  
166         address operator,  
167         uint256 id  
168     ) external {  
169         address owner = _ownerOf(id);  
170         require(sender != address(0), "sender is zero address");  
171         require(  
172             msg.sender == sender ||  
173             _metaTransactionContracts[msg.sender] ||  
174             _superOperators[msg.sender] ||  
175             _operatorsForAll[sender][msg.sender],  
176             "not authorized to approve"
```

```

177     );
178     require(owner == sender, "owner != sender");
179     _approveFor(owner, operator, id);
180 }

```

✓ The code meets the specification.

Formal Verification Request 148

Method will not encounter an assertion failure.



10, Dec 2019



5.04 ms

Line 148 in File ERC721BaseToken.sol

```

148 // @CTK NO_ASF

```

Line 164-180 in File ERC721BaseToken.sol

```

164 function approveFor(
165     address sender,
166     address operator,
167     uint256 id
168 ) external {
169     address owner = _ownerOf(id);
170     require(sender != address(0), "sender is zero address");
171     require(
172         msg.sender == sender ||
173         _metaTransactionContracts[msg.sender] ||
174         _superOperators[msg.sender] ||
175         _operatorsForAll[sender][msg.sender],
176         "not authorized to approve"
177     );
178     require(owner == sender, "owner != sender");
179     _approveFor(owner, operator, id);
180 }

```

✓ The code meets the specification.

Formal Verification Request 149

approveFor_require



10, Dec 2019



12.15 ms

Line 149-154 in File ERC721BaseToken.sol

```

149 /* @CTK approveFor_require
150    @tag assume_completion
151    @post sender != address(0)
152    @post sender == address(_owners[id])
153    @post (msg.sender == sender) || (_metaTransactionContracts[msg.sender]) || (
154        _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])

```


Line 164-180 in File ERC721BaseToken.sol

```

164     function approveFor(
165         address sender,
166         address operator,
167         uint256 id
168     ) external {
169         address owner = _ownerOf(id);
170         require(sender != address(0), "sender is zero address");
171         require(
172             msg.sender == sender ||
173             _metaTransactionContracts[msg.sender] ||
174             _superOperators[msg.sender] ||
175             _operatorsForAll[sender][msg.sender],
176             "not authorized to approve"
177         );
178         require(owner == sender, "owner != sender");
179         _approveFor(owner, operator, id);
180     }

```

✓ The code meets the specification.

Formal Verification Request 150

approveFor_change



10, Dec 2019



3.24 ms

Line 155-163 in File ERC721BaseToken.sol

```

155     /*@CTK approveFor_change
156     @tag assume_completion
157     @pre sender != address(0)
158     @pre sender == address(_owners[id])
159     @pre (msg.sender == sender) || (_metaTransactionContracts[msg.sender]) || (
160         _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])
161     @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
162     @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
163         2**255)
164     @post (operator != address(0)) -> (__post._operators[id] == operator)
165     */

```

Line 164-180 in File ERC721BaseToken.sol

```

164     function approveFor(
165         address sender,
166         address operator,
167         uint256 id
168     ) external {
169         address owner = _ownerOf(id);
170         require(sender != address(0), "sender is zero address");
171         require(
172             msg.sender == sender ||
173             _metaTransactionContracts[msg.sender] ||
174             _superOperators[msg.sender] ||
175             _operatorsForAll[sender][msg.sender],
176             "not authorized to approve"

```

```

177     );
178     require(owner == sender, "owner != sender");
179     _approveFor(owner, operator, id);
180 }


```

✓ The code meets the specification.

Formal Verification Request 151

If method completes, integer overflow would not happen.

 10, Dec 2019

 54.66 ms

Line 187 in File ERC721BaseToken.sol

```

187 // @CTK NO_OVERFLOW

```

Line 203-213 in File ERC721BaseToken.sol

```

203 function approve(address operator, uint256 id) external {
204     address owner = _ownerOf(id);
205     require(owner != address(0), "token does not exist");
206     require(
207         owner == msg.sender ||
208         _superOperators[msg.sender] ||
209         _operatorsForAll[owner][msg.sender],
210         "not authorized to approve"
211     );
212     _approveFor(owner, operator, id);
213 }


```

✓ The code meets the specification.

Formal Verification Request 152

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

 10, Dec 2019

 3.74 ms

Line 188 in File ERC721BaseToken.sol

```

188 // @CTK NO_BUF_OVERFLOW

```

Line 203-213 in File ERC721BaseToken.sol

```

203 function approve(address operator, uint256 id) external {
204     address owner = _ownerOf(id);
205     require(owner != address(0), "token does not exist");
206     require(
207         owner == msg.sender ||
208         _superOperators[msg.sender] ||
209         _operatorsForAll[owner][msg.sender],
210         "not authorized to approve"
211     );
212     _approveFor(owner, operator, id);
213 }

```

✓ The code meets the specification.

Formal Verification Request 153

Method will not encounter an assertion failure.

10, Dec 2019

3.74 ms

Line 189 in File ERC721BaseToken.sol

189 `//@CTK NO_ASF`

Line 203-213 in File ERC721BaseToken.sol

```
203     function approve(address operator, uint256 id) external {
204         address owner = _ownerOf(id);
205         require(owner != address(0), "token does not exist");
206         require(
207             owner == msg.sender ||
208             _superOperators[msg.sender] ||
209             _operatorsForAll[owner][msg.sender],
210             "not authorized to approve"
211         );
212         _approveFor(owner, operator, id);
213     }
```

✓ The code meets the specification.

Formal Verification Request 154

approve__require

10, Dec 2019

3.41 ms

Line 190-194 in File ERC721BaseToken.sol

```
190     /*@CTK approve_require
191         @tag assume_completion
192         @post address(_owners[id]) != address(0)
193         @post (msg.sender == address(_owners[id])) || (_superOperators[msg.sender]) || (
194             _operatorsForAll[address(_owners[id])][msg.sender])
195     */
```

Line 203-213 in File ERC721BaseToken.sol

```
203     function approve(address operator, uint256 id) external {
204         address owner = _ownerOf(id);
205         require(owner != address(0), "token does not exist");
206         require(
207             owner == msg.sender ||
208             _superOperators[msg.sender] ||
209             _operatorsForAll[owner][msg.sender],
210             "not authorized to approve"
211         );
212         _approveFor(owner, operator, id);
213     }
```

✓ The code meets the specification.

Formal Verification Request 155

approve_change

10, Dec 2019

3.48 ms

Line 195-202 in File ERC721BaseToken.sol

```

195  /*@CTK approve_change
196      @tag assume_completion
197      @pre address(_owners[id]) != address(0)
198      @pre (msg.sender == address(_owners[id])) || (_superOperators[msg.sender]) || (
          _operatorsForAll[address(_owners[id])][msg.sender])
199      @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
200      @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
          2**255)
201      @post (operator != address(0)) -> (__post._operators[id] == operator)
202  */

```

Line 203-213 in File ERC721BaseToken.sol

```

203  function approve(address operator, uint256 id) external {
204      address owner = _ownerOf(id);
205      require(owner != address(0), "token does not exist");
206      require(
207          owner == msg.sender ||
208          _superOperators[msg.sender] ||
209          _operatorsForAll[owner][msg.sender],
210          "not authorized to approve"
211      );
212      _approveFor(owner, operator, id);
213  }

```

✓ The code meets the specification.

Formal Verification Request 156

If method completes, integer overflow would not happen.

10, Dec 2019

28.57 ms

Line 220 in File ERC721BaseToken.sol

```

220  //@CTK NO_OVERFLOW

```

Line 229-237 in File ERC721BaseToken.sol

```

229  function getApproved(uint256 id) external view returns (address) {
230      (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
231      require(owner != address(0), "token does not exist");
232      if (operatorEnabled) {
233          return _operators[id];
234      } else {

```

```
235         return address(0);
236     }
237 }
```

✓ The code meets the specification.

Formal Verification Request 157

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

📅 10, Dec 2019

🕒 0.49 ms

Line 221 in File ERC721BaseToken.sol

```
221 // @CTK NO_BUF_OVERFLOW
```

Line 229-237 in File ERC721BaseToken.sol

```
229 function getApproved(uint256 id) external view returns (address) {
230     (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
231     require(owner != address(0), "token does not exist");
232     if (operatorEnabled) {
233         return _operators[id];
234     } else {
235         return address(0);
236     }
237 }
```

✓ The code meets the specification.

Formal Verification Request 158

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 10.25 ms

Line 222 in File ERC721BaseToken.sol

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

Line 229-237 in File ERC721BaseToken.sol

```
229 function getApproved(uint256 id) external view returns (address) {
230     (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
231     require(owner != address(0), "token does not exist");
232     if (operatorEnabled) {
233         return _operators[id];
234     } else {
235         return address(0);
236     }
237 }
```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     id = 0
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    __return = 0
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25  }
26  Address_Map = [
27    {
28      "key": "ALL_OTHERS",
29      "value": {
30        "contract_name": "ERC721BaseToken",
31        "balance": 0,
32        "contract": {
33          "_ERC721_RECEIVED": "AAAA",
34          "_ERC721_BATCH_RECEIVED": "AAAA",
35          "ERC165ID": "AAAA",
36          "ERC721_MANDATORY_RECEIVER": "\u0081\u0081\u0081\u0081",
37          "_numNFTPerAddress": [
38            {
39              "key": 0,
40              "value": 64
41            },
42            {
43              "key": 4,
44              "value": 8
45            },
46            {
47              "key": 1,
48              "value": 16
49            },
50            {
51              "key": "ALL_OTHERS",
52              "value": 0
53            }
54          ],
55          "_owners": [
56            {
57              "key": 128,
58              "value": 16

```

```

59         },
60         {
61             "key": 2,
62             "value": 32
63         },
64         {
65             "key": "ALL_OTHERS",
66             "value": 0
67         }
68     ],
69     "_operatorsForAll": [
70         {
71             "key": "ALL_OTHERS",
72             "value": [
73                 {
74                     "key": "ALL_OTHERS",
75                     "value": false
76                 }
77             ]
78         }
79     ],
80     "_operators": [
81         {
82             "key": 16,
83             "value": 128
84         },
85         {
86             "key": 8,
87             "value": 32
88         },
89         {
90             "key": "ALL_OTHERS",
91             "value": 0
92         }
93     ],
94     "_metaTransactionContracts": [
95         {
96             "key": "ALL_OTHERS",
97             "value": false
98         }
99     ],
100     "_admin": 0,
101     "_superOperators": [
102         {
103             "key": "ALL_OTHERS",
104             "value": true
105         }
106     ]
107 }
108 }
109 }
110 ]

```

112 Function invocation is reverted.

Formal Verification Request 159

getApproved

📅 10, Dec 2019

🕒 0.51 ms

Line 223-228 in File ERC721BaseToken.sol

```

223  /*@CTK getApproved
224  @tag assume_completion
225  @post address(_owners[id]) != address(0)
226  @post ((_owners[id] / 2**255) == 1) -> (__return == _operators[id])
227  @post ((_owners[id] / 2**255) != 1) -> (__return == address(0))
228  */

```

Line 229-237 in File ERC721BaseToken.sol

```

229  function getApproved(uint256 id) external view returns (address) {
230      (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
231      require(owner != address(0), "token does not exist");
232      if (operatorEnabled) {
233          return _operators[id];
234      } else {
235          return address(0);
236      }
237  }

```

✅ The code meets the specification.

Formal Verification Request 160

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 52.39 ms

Line 239 in File ERC721BaseToken.sol

```

239  //@CTK NO_OVERFLOW

```

Line 249-263 in File ERC721BaseToken.sol

```

249  function _checkTransfer(address from, address to, uint256 id) internal view returns (
250      bool isMetaTx) {
251      (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
252      require(owner != address(0), "token does not exist");
253      require(owner == from, "not owner in _checkTransfer");
254      require(to != address(0), "can't send to zero address");
255      isMetaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
256      if (msg.sender != from && !isMetaTx) {
257          require(
258              _superOperators[msg.sender] ||
259              _operatorsForAll[from][msg.sender] ||
260              (operatorEnabled && _operators[id] == msg.sender),
261              "not approved to transfer"
262          );
263      }

```


✓ The code meets the specification.

Formal Verification Request 161

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

📅 10, Dec 2019

🕒 0.66 ms

Line 240 in File ERC721BaseToken.sol

240 `//@CTK NO_BUF_OVERFLOW`

Line 249-263 in File ERC721BaseToken.sol

```
249     function _checkTransfer(address from, address to, uint256 id) internal view returns (
        bool isMetaTx) {
250         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
251         require(owner != address(0), "token does not exist");
252         require(owner == from, "not owner in _checkTransfer");
253         require(to != address(0), "can't send to zero address");
254         isMetaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
255         if (msg.sender != from && !isMetaTx) {
256             require(
257                 _superOperators[msg.sender] ||
258                 _operatorsForAll[from][msg.sender] ||
259                 (operatorEnabled && _operators[id] == msg.sender),
260                 "not approved to transfer"
261             );
262         }
263     }
```

✓ The code meets the specification.

Formal Verification Request 162

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 9.76 ms

Line 241 in File ERC721BaseToken.sol

241 `//@CTK FAIL NO_ASF`

Line 249-263 in File ERC721BaseToken.sol

```
249     function _checkTransfer(address from, address to, uint256 id) internal view returns (
        bool isMetaTx) {
250         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
251         require(owner != address(0), "token does not exist");
252         require(owner == from, "not owner in _checkTransfer");
253         require(to != address(0), "can't send to zero address");
254         isMetaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
255         if (msg.sender != from && !isMetaTx) {
256             require(
257                 _superOperators[msg.sender] ||
```

```

258         _operatorsForAll[from][msg.sender] ||
259         (operatorEnabled && _operators[id] == msg.sender),
260         "not approved to transfer"
261     );
262 }
263 }

```

✖ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     from = 0
5     id = 0
6     to = 0
7   }
8   This = 0
9   Internal = {
10     __has_assertion_failure = false
11     __has_buf_overflow = false
12     __has_overflow = false
13     __has_returned = false
14     __reverted = false
15     msg = {
16       "gas": 0,
17       "sender": 0,
18       "value": 0
19     }
20   }
21   Other = {
22     block = {
23       "number": 0,
24       "timestamp": 0
25     }
26     isMetaTx = false
27   }
28   Address_Map = [
29     {
30       "key": "ALL_OTHERS",
31       "value": {
32         "contract_name": "ERC721BaseToken",
33         "balance": 0,
34         "contract": {
35           "_ERC721_RECEIVED": "AAAA",
36           "_ERC721_BATCH_RECEIVED": "AAAA",
37           "ERC165ID": "QQQQ",
38           "ERC721_MANDATORY_RECEIVER": "AAAA",
39           "_numNFTPerAddress": [
40             {
41               "key": 8,
42               "value": 64
43             },
44             {
45               "key": 4,
46               "value": 8
47             },
48             {
49               "key": "ALL_OTHERS",
50               "value": 0

```

```

51     }
52 ],
53 "_owners": [
54     {
55         "key": 2,
56         "value": 32
57     },
58     {
59         "key": 16,
60         "value": 2
61     },
62     {
63         "key": "ALL_OTHERS",
64         "value": 0
65     }
66 ],
67 "_operatorsForAll": [
68     {
69         "key": "ALL_OTHERS",
70         "value": [
71             {
72                 "key": "ALL_OTHERS",
73                 "value": false
74             }
75         ]
76     }
77 ],
78 "_operators": [
79     {
80         "key": 4,
81         "value": 64
82     },
83     {
84         "key": 0,
85         "value": 32
86     },
87     {
88         "key": 64,
89         "value": 4
90     },
91     {
92         "key": 16,
93         "value": 128
94     },
95     {
96         "key": "ALL_OTHERS",
97         "value": 0
98     }
99 ],
100 "_metaTransactionContracts": [
101     {
102         "key": "ALL_OTHERS",
103         "value": true
104     }
105 ],
106 "_admin": 0,
107 "_superOperators": [
108     {

```

```

109         "key": "ALL_OTHERS",
110         "value": false
111     }
112 }
113 }
114 }
115 }
116 ]
117
118 Function invocation is reverted.

```

Formal Verification Request 163

`__checkTransfer`



10, Dec 2019



0.74 ms

Line 242-248 in File ERC721BaseToken.sol

```

242  /*@CTK __checkTransfer
243     @tag assume_completion
244     @post address(_owners[id]) != address(0)
245     @post from == _owners[id]
246     @post to != address(0)
247     @post (msg.sender != from) && (_metaTransactionContracts[msg.sender] == false) ->
        _superOperators[msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id]
        ] / 2**255) == 1) && _operators[id] == msg.sender)
248  */

```

Line 249-263 in File ERC721BaseToken.sol

```

249  function __checkTransfer(address from, address to, uint256 id) internal view returns (
    bool isMetaTx) {
250      (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
251      require(owner != address(0), "token does not exist");
252      require(owner == from, "not owner in __checkTransfer");
253      require(to != address(0), "can't send to zero address");
254      isMetaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
255      if (msg.sender != from && !isMetaTx) {
256          require(
257              _superOperators[msg.sender] ||
258              _operatorsForAll[from][msg.sender] ||
259              (operatorEnabled && _operators[id] == msg.sender),
260              "not approved to transfer"
261          );
262      }
263  }

```

✓ The code meets the specification.

Formal Verification Request 164

If method completes, integer overflow would not happen.



10, Dec 2019



6.26 ms

Line 265 in File ERC721BaseToken.sol

265 `//@CTK NO_OVERFLOW`

Line 272-314 in File ERC721BaseToken.sol

```

272     function _checkInterfaceWith10000Gas(address _contract, bytes4 interfaceId)
273         internal
274         view
275         returns (bool)
276     {
277         bool success;
278         bool result;
279         bytes memory call_data = abi.encodeWithSelector(
280             ERC165ID,
281             interfaceId
282         );
283         // solium-disable-next-line security/no-inline-assembly
284         /*@CTK _checkInterfaceWith10000Gas_assembly
285             @tag assume_completion
286             @var bool success
287             @var bool result
288             @post result == true
289             @post success == true
290         */
291         // solium-disable-next-line security/no-inline-assembly
292         assembly {
293             let call_ptr := add(0x20, call_data)
294             let call_size := mload(call_data)
295             let output := mload(0x40) // Find empty storage location using "free memory
                pointer"
296             mstore(output, 0x0)
297             success := staticcall(
298                 10000,
299                 _contract,
300                 call_ptr,
301                 call_size,
302                 output,
303                 0x20
304             ) // 32 bytes
305             result := mload(output)
306         }
307         // (10000 / 63) "not enough for supportsInterface(...)" // consume all gas, so
            caller can potentially know that there was not enough gas
308         assert(gasleft() > 158);
309         return success && result;
310     }

```

✓ The code meets the specification.

Formal Verification Request 165

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



10, Dec 2019



0.27 ms

Line 266 in File ERC721BaseToken.sol

```
266 // @CTK NO_BUF_OVERFLOW
```

Line 272-314 in File ERC721BaseToken.sol

```
272 function _checkInterfaceWith10000Gas(address _contract, bytes4 interfaceId)
273     internal
274     view
275     returns (bool)
276 {
277     bool success;
278     bool result;
279     bytes memory call_data = abi.encodeWithSelector(
280         ERC165ID,
281         interfaceId
282     );
283     // solium-disable-next-line security/no-inline-assembly
284     /* @CTK _checkInterfaceWith10000Gas_assembly
285         @tag assume_completion
286         @var bool success
287         @var bool result
288         @post result == true
289         @post success == true
290     */
291     // solium-disable-next-line security/no-inline-assembly
292     assembly {
293         let call_ptr := add(0x20, call_data)
294         let call_size := mload(call_data)
295         let output := mload(0x40) // Find empty storage location using "free memory
296             pointer"
297         mstore(output, 0x0)
298         success := staticcall(
299             10000,
300             _contract,
301             call_ptr,
302             call_size,
303             output,
304             0x20
305         ) // 32 bytes
306         result := mload(output)
307     }
308     // (10000 / 63) "not enough for supportsInterface(...)" // consume all gas, so
309     // caller can potentially know that there was not enough gas
310     assert(gasleft() > 158);
311     return success && result;
312 }
```

✓ The code meets the specification.

Formal Verification Request 166

Method will not encounter an assertion failure.



10, Dec 2019



0.27 ms

Line 267 in File ERC721BaseToken.sol

267 `//@CTK NO_ASF`

Line 272-314 in File ERC721BaseToken.sol

```

272 function _checkInterfaceWith10000Gas(address _contract, bytes4 interfaceId)
273     internal
274     view
275     returns (bool)
276 {
277     bool success;
278     bool result;
279     bytes memory call_data = abi.encodeWithSelector(
280         ERC165ID,
281         interfaceId
282     );
283     // solium-disable-next-line security/no-inline-assembly
284     /*@CTK _checkInterfaceWith10000Gas_assembly
285         @tag assume_completion
286         @var bool success
287         @var bool result
288         @post result == true
289         @post success == true
290     */
291     // solium-disable-next-line security/no-inline-assembly
292     assembly {
293         let call_ptr := add(0x20, call_data)
294         let call_size := mload(call_data)
295         let output := mload(0x40) // Find empty storage location using "free memory
296                                 // pointer"
297         mstore(output, 0x0)
298         success := staticcall(
299             10000,
300             _contract,
301             call_ptr,
302             call_size,
303             output,
304             0x20
305         ) // 32 bytes
306         result := mload(output)
307     }
308     // (10000 / 63) "not enough for supportsInterface(...)" // consume all gas, so
309     // caller can potentially know that there was not enough gas
310     assert(gasleft() > 158);
311     return success && result;
312 }

```

✓ The code meets the specification.

Formal Verification Request 167

`_checkInterfaceWith10000Gas`



10, Dec 2019



0.29 ms

Line 268-271 in File ERC721BaseToken.sol

```
268 /*@CTK _checkInterfaceWith10000Gas
269 @tag assume_completion
270 @post __return == true
271 */
```

Line 272-314 in File ERC721BaseToken.sol


```
272 function _checkInterfaceWith10000Gas(address _contract, bytes4 interfaceId)
273     internal
274     view
275     returns (bool)
276 {
277     bool success;
278     bool result;
279     bytes memory call_data = abi.encodeWithSelector(
280         ERC165ID,
281         interfaceId
282     );
283     // solium-disable-next-line security/no-inline-assembly
284     /*@CTK _checkInterfaceWith10000Gas_assembly
285     @tag assume_completion
286     @var bool success
287     @var bool result
288     @post result == true
289     @post success == true
290     */
291     // solium-disable-next-line security/no-inline-assembly
292     assembly {
293         let call_ptr := add(0x20, call_data)
294         let call_size := mload(call_data)
295         let output := mload(0x40) // Find empty storage location using "free memory
296             pointer"
297         mstore(output, 0x0)
298         success := staticcall(
299             10000,
300             _contract,
301             call_ptr,
302             call_size,
303             output,
304             0x20
305         ) // 32 bytes
306         result := mload(output)
307     }
308     // (10000 / 63) "not enough for supportsInterface(...)" // consume all gas, so
309     // caller can potentially know that there was not enough gas
310     assert(gasleft() > 158);
311     return success && result;
312 }
```

✓ The code meets the specification.

Formal Verification Request 168

If method completes, integer overflow would not happen.

 10, Dec 2019

 75.49 ms

Line 322 in File ERC721BaseToken.sol

322 `//@CTK NO_OVERFLOW`


Line 334-345 in File ERC721BaseToken.sol

```
334     function transferFrom(address from, address to, uint256 id) external {
335         bool metaTx = _checkTransfer(from, to, id);
336         _transferFrom(from, to, id);
337         if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
338             require(
339                 _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, ""),
340                 "erc721 transfer rejected by to"
341             );
342         }
343     }
```

 The code meets the specification.

Formal Verification Request 169

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

 10, Dec 2019 1.1 ms

Line 323 in File ERC721BaseToken.sol

323 `//@CTK NO_BUF_OVERFLOW`


Line 334-345 in File ERC721BaseToken.sol

```
334     function transferFrom(address from, address to, uint256 id) external {
335         bool metaTx = _checkTransfer(from, to, id);
336         _transferFrom(from, to, id);
337         if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
338             require(
339                 _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, ""),
340                 "erc721 transfer rejected by to"
341             );
342         }
343     }
```

 The code meets the specification.

Formal Verification Request 170

Method will not encounter an assertion failure.

 10, Dec 2019 10.04 ms

Line 324 in File ERC721BaseToken.sol

324 `//@CTK FAIL NO_ASF`

Line 334-345 in File ERC721BaseToken.sol

```

334 function transferFrom(address from, address to, uint256 id) external {
335     bool metaTx = _checkTransfer(from, to, id);
336     _transferFrom(from, to, id);
337     if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
338         require(
339             _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, ""),
340             "erc721 transfer rejected by to"
341         );
342     }
343 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4       from = 0
5       id = 0
6       to = 0
7   }
8   This = 0
9   Internal = {
10       __has_assertion_failure = false
11       __has_buf_overflow = false
12       __has_overflow = false
13       __has_returned = false
14       __reverted = false
15       msg = {
16           "gas": 0,
17           "sender": 0,
18           "value": 0
19       }
20   }
21   Other = {
22       block = {
23           "number": 0,
24           "timestamp": 0
25       }
26   }
27   Address_Map = [
28       {
29           "key": "ALL_OTHERS",
30           "value": {
31               "contract_name": "ERC721BaseToken",
32               "balance": 0,
33               "contract": {
34                   "_ERC721_RECEIVED": "AAAA",
35                   "_ERC721_BATCH_RECEIVED": "AAAA",
36                   "ERC165ID": "EEEE",
37                   "ERC721_MANDATORY_RECEIVER": "AAAA",
38                   "_numNFTPerAddress": [
39                       {
40                           "key": 4,
41                           "value": 8
42                       },
43                       {
44                           "key": 8,
45                           "value": 64

```

```

46     },
47     {
48         "key": "ALL_OTHERS",
49         "value": 0
50     }
51 ],
52 "_owners": [
53     {
54         "key": 16,
55         "value": 2
56     },
57     {
58         "key": 2,
59         "value": 32
60     },
61     {
62         "key": "ALL_OTHERS",
63         "value": 0
64     }
65 ],
66 "_operatorsForAll": [
67     {
68         "key": "ALL_OTHERS",
69         "value": [
70             {
71                 "key": "ALL_OTHERS",
72                 "value": true
73             }
74         ]
75     }
76 ],
77 "_operators": [
78     {
79         "key": 16,
80         "value": 128
81     },
82     {
83         "key": 0,
84         "value": 32
85     },
86     {
87         "key": "ALL_OTHERS",
88         "value": 0
89     }
90 ],
91 "_metaTransactionContracts": [
92     {
93         "key": "ALL_OTHERS",
94         "value": false
95     }
96 ],
97 "_admin": 0,
98 "_superOperators": [
99     {
100         "key": "ALL_OTHERS",
101         "value": true
102     }
103 ]

```

```

104     }
105   }
106 }
107 ]
108
109 Function invocation is reverted.

```

Formal Verification Request 171

transferFrom



10, Dec 2019



1.32 ms

Line 325-333 in File ERC721BaseToken.sol

```

325  /*@CTK transferFrom
326    @tag assume_completion
327    @pre (from == _owners[id]) && (from != address(0))
328    @pre to != 0
329    @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || _superOperators[
      msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id] / 2**255) ==
      1) && _operators[id] == msg.sender)
330    @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
331    @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
332    @post __post._owners[id] == uint256(to)
333  */

```

Line 334-345 in File ERC721BaseToken.sol

```

334  function transferFrom(address from, address to, uint256 id) external {
335    bool metaTx = _checkTransfer(from, to, id);
336    _transferFrom(from, to, id);
337    if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
338      require(
339        _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, ""),
340        "erc721 transfer rejected by to"
341      );
342    }
343  }

```

✓ The code meets the specification.

Formal Verification Request 172

If method completes, integer overflow would not happen.



10, Dec 2019



75.16 ms

Line 354 in File ERC721BaseToken.sol

```

354  /*@CTK NO_OVERFLOW

```

Line 366-377 in File ERC721BaseToken.sol

```
366     function safeTransferFrom(address from, address to, uint256 id, bytes memory data)
367     public {
368         bool metaTx = _checkTransfer(from, to, id);
369         _transferFrom(from, to, id);
370         if (to.isContract()) {
371             require(
372                 _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, data),
373                 "ERC721: transfer rejected by to"
374             );
375         }
376     }
```

✓ The code meets the specification.

Formal Verification Request 173

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

📅 10, Dec 2019

🕒 1.13 ms

Line 355 in File ERC721BaseToken.sol

```
355     //@CTK_NO_BUF_OVERFLOW
```

Line 366-377 in File ERC721BaseToken.sol

```
366     function safeTransferFrom(address from, address to, uint256 id, bytes memory data)
367     public {
368         bool metaTx = _checkTransfer(from, to, id);
369         _transferFrom(from, to, id);
370         if (to.isContract()) {
371             require(
372                 _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, data),
373                 "ERC721: transfer rejected by to"
374             );
375         }
376     }
```

✓ The code meets the specification.

Formal Verification Request 174

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 9.3 ms

Line 356 in File ERC721BaseToken.sol

```
356     //@CTK_FAIL_NO_ASF
```

Line 366-377 in File ERC721BaseToken.sol

```
366     function safeTransferFrom(address from, address to, uint256 id, bytes memory data)
367     public {
368         bool metaTx = _checkTransfer(from, to, id);
```

```

368     _transferFrom(from, to, id);
369     if (to.isContract()) {
370         require(
371             _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, data),
372             "ERC721: transfer rejected by to"
373         );
374     }
375 }

```

✖ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     data = ""
5     from = 0
6     id = 0
7     to = 0
8   }
9   This = 0
10  Internal = {
11    __has_assertion_failure = false
12    __has_buf_overflow = false
13    __has_overflow = false
14    __has_returned = false
15    __reverted = false
16    msg = {
17      "gas": 0,
18      "sender": 0,
19      "value": 0
20    }
21  }
22  Other = {
23    block = {
24      "number": 0,
25      "timestamp": 0
26    }
27  }
28  Address_Map = [
29    {
30      "key": "ALL_OTHERS",
31      "value": {
32        "contract_name": "ERC721BaseToken",
33        "balance": 0,
34        "contract": {
35          "_ERC721_RECEIVED": "AAAA",
36          "_ERC721_BATCH_RECEIVED": "AAAA",
37          "ERC165ID": "QQQQ",
38          "ERC721_MANDATORY_RECEIVER": "AAAA",
39          "_numNFTPerAddress": [
40            {
41              "key": 0,
42              "value": 0
43            },
44            {
45              "key": 8,
46              "value": 64
47            },
48            {

```

```

49         "key": 64,
50         "value": 0
51     },
52     {
53         "key": "ALL_OTHERS",
54         "value": 8
55     }
56 ],
57 "_owners": [
58     {
59         "key": 2,
60         "value": 32
61     },
62     {
63         "key": 16,
64         "value": 2
65     },
66     {
67         "key": "ALL_OTHERS",
68         "value": 0
69     }
70 ],
71 "_operatorsForAll": [
72     {
73         "key": "ALL_OTHERS",
74         "value": [
75             {
76                 "key": "ALL_OTHERS",
77                 "value": false
78             }
79         ]
80     }
81 ],
82 "_operators": [
83     {
84         "key": 0,
85         "value": 32
86     },
87     {
88         "key": 4,
89         "value": 64
90     },
91     {
92         "key": 64,
93         "value": 4
94     },
95     {
96         "key": 16,
97         "value": 128
98     },
99     {
100         "key": "ALL_OTHERS",
101         "value": 0
102     }
103 ],
104 "_metaTransactionContracts": [
105     {
106         "key": "ALL_OTHERS",

```

```


107         "value": false
108     }
109 ],
110     "_admin": 0,
111     "_superOperators": [
112     {
113         "key": "ALL_OTHERS",
114         "value": true
115     }
116 ]
117 }
118 }
119 }
120 ]
121
122 Function invocation is reverted.

```

Formal Verification Request 175

safeTransferFrom

 10, Dec 2019

 1.46 ms

Line 357-365 in File ERC721BaseToken.sol

```

357     /*@CTK safeTransferFrom
358     @tag assume_completion
359     @pre (from == _owners[id]) && (from != address(0))
360     @pre to != address(0)
361     @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || _superOperators[
        msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id] / 2**255) ==
        1) && _operators[id] == msg.sender)
362     @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
363     @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
364     @post __post._owners[id] == uint256(to)
365     */

```

Line 366-377 in File ERC721BaseToken.sol

```

366     function safeTransferFrom(address from, address to, uint256 id, bytes memory data)
        public {
367         bool metaTx = _checkTransfer(from, to, id);
368         _transferFrom(from, to, id);
369         if (to.isContract()) {
370             require(
371                 _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, data),
372                 "ERC721: transfer rejected by to"
373             );
374         }
375     }

```

 The code meets the specification.

Formal Verification Request 176

If method completes, integer overflow would not happen.

10, Dec 2019

2150.61 ms

Line 400 in File ERC721BaseToken.sol

```
400 // @CTK FAIL NO_OVERFLOW
```

Line 409-458 in File ERC721BaseToken.sol

```

409 function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory
    data, bool safe) internal {
410     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
411     bool authorized = msg.sender == from ||
412         metaTx ||
413         _superOperators[msg.sender] ||
414         _operatorsForAll[from][msg.sender];
415
416     require(from != address(0), "from is zero address");
417     require(to != address(0), "can't send to zero address");
418
419     uint256 numTokens = ids.length;
420     /* @CTK "_batchTransferFrom_loop"
421        @pre from != address(0)
422        @pre to != address(0)
423        @pre numTokens < 5
424        @inv this._numNFTPerAddress[from] == this._pre._numNFTPerAddress[from]
425        @inv this._numNFTPerAddress[to] == this._pre._numNFTPerAddress[to]
426        @inv ids == ids__pre
427        @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from
428            )
429        @pre forall j: uint. (j >= 0 /\ j < numTokens) -> ((msg.sender == from) || (this.
430            _metaTransactionContracts[msg.sender]) || (this._superOperators[msg.sender]) ||
431            (this._operatorsForAll[from][msg.sender])) || (((this._owners[ids[j]] /
432            2**255) == 1) && (this._operators[ids[j]] == msg.sender))
433        @inv i <= numTokens
434        @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
435        @inv numTokens == numTokens__pre
436        @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this._pre.
437            _numNFTPerAddress[from] + this._pre._numNFTPerAddress[to])
438        @post this._numNFTPerAddress[from] == this._pre._numNFTPerAddress[from]
439        @post this._numNFTPerAddress[to] == this._pre._numNFTPerAddress[to]
440        @post i == numTokens
441        @post !_should_return
442        */
443     for(uint256 i = 0; i < numTokens; i++) {
444         uint256 id = ids[i];
445         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
446         require(owner == from, "not owner in batchTransferFrom");
447         require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not
            authorized");
448         _owners[id] = uint256(to);
449         // emit Transfer(from, to, id);
450     }
451     if (from != to) {
452         _numNFTPerAddress[from] -= numTokens;

```

```

448     _numNFTPerAddress[to] += numTokens;
449 }
450 if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to,
    ERC721_MANDATORY_RECEIVER))) {
451     require(
452         _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
453         "erc721 batch transfer rejected by to"
454     );
455 }
456 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     data = ""
5     from = 2
6     ids = [
7       0,
8       0,
9       0,
10      0
11    ]
12    safe = false
13    to = 16
14  }
15  This = 0
16  Internal = {
17    __has_assertion_failure = false
18    __has_buf_overflow = false
19    __has_overflow = false
20    __has_returned = false
21    __reverted = false
22    msg = {
23      "gas": 0,
24      "sender": 0,
25      "value": 0
26    }
27  }
28  Other = {
29    block = {
30      "number": 0,
31      "timestamp": 0
32    }
33  }
34  Address_Map = [
35    {
36      "key": 0,
37      "value": {
38        "contract_name": "ERC721BaseToken",
39        "balance": 0,
40        "contract": {
41          "_ERC721_RECEIVED": "AAAA",
42          "_ERC721_BATCH_RECEIVED": "AAAA",
43          "ERC165ID": "AAAA",
44          "ERC721_MANDATORY_RECEIVER": "AAAA",
45          "_numNFTPerAddress": [
46

```

```

47     "key": 0,
48     "value": 4
49 },
50 {
51     "key": 128,
52     "value": 64
53 },
54 {
55     "key": "ALL_OTHERS",
56     "value": 0
57 }
58 ],
59 "_owners": [
60 {
61     "key": "ALL_OTHERS",
62     "value": 0
63 }
64 ],
65 "_operatorsForAll": [
66 {
67     "key": "ALL_OTHERS",
68     "value": [
69         {
70             "key": "ALL_OTHERS",
71             "value": false
72         }
73     ]
74 }
75 ],
76 "_operators": [
77 {
78     "key": 1,
79     "value": 1
80 },
81 {
82     "key": 0,
83     "value": 4
84 },
85 {
86     "key": "ALL_OTHERS",
87     "value": 0
88 }
89 ],
90 "_metaTransactionContracts": [
91 {
92     "key": 32,
93     "value": true
94 },
95 {
96     "key": "ALL_OTHERS",
97     "value": false
98 }
99 ],
100 "_admin": 0,
101 "_superOperators": [
102 {
103     "key": "ALL_OTHERS",
104     "value": false

```

```

105     }
106   ]
107 }
108 }
109 },
110 {
111   "key": "ALL_OTHERS",
112   "value": "EmptyAddress"
113 }
114 ]
115
116 After Execution:
117   Input = {
118     data = ""
119     from = 4
120     ids = [
121       0,
122       0,
123       0,
124       0
125     ]
126     safe = false
127     to = 0
128   }
129   This = 0
130   Internal = {
131     __has_assertion_failure = false
132     __has_buf_overflow = false
133     __has_overflow = true
134     __has_returned = false
135     __reverted = false
136     msg = {
137       "gas": 0,
138       "sender": 0,
139       "value": 0
140     }
141   }
142   Other = {
143     block = {
144       "number": 0,
145       "timestamp": 0
146     }
147   }
148   Address_Map = [
149     {
150       "key": 0,
151       "value": {
152         "contract_name": "ERC721BaseToken",
153         "balance": 0,
154         "contract": {
155           "_ERC721_RECEIVED": "AAAA",
156           "_ERC721_BATCH_RECEIVED": "AAAA",
157           "ERC165ID": "AAAA",
158           "ERC721_MANDATORY_RECEIVER": "AAAA",
159           "_numNFTPerAddress": [
160             {
161               "key": 4,
162               "value": 252

```

```

163     },
164     {
165         "key": 0,
166         "value": 8
167     },
168     {
169         "key": 128,
170         "value": 64
171     },
172     {
173         "key": "ALL_OTHERS",
174         "value": 0
175     }
176 ],
177 "_owners": [
178     {
179         "key": "ALL_OTHERS",
180         "value": 0
181     }
182 ],
183 "_operatorsForAll": [
184     {
185         "key": "ALL_OTHERS",
186         "value": [
187             {
188                 "key": "ALL_OTHERS",
189                 "value": false
190             }
191         ]
192     }
193 ],
194 "_operators": [
195     {
196         "key": 1,
197         "value": 1
198     },
199     {
200         "key": 0,
201         "value": 4
202     },
203     {
204         "key": "ALL_OTHERS",
205         "value": 0
206     }
207 ],
208 "_metaTransactionContracts": [
209     {
210         "key": 32,
211         "value": true
212     },
213     {
214         "key": "ALL_OTHERS",
215         "value": false
216     }
217 ],
218 "_admin": 0,
219 "_superOperators": [
220     {

```

```


221         "key": "ALL_OTHERS",
222         "value": false
223     }
224 ]
225 }
226 }
227 },
228 {
229     "key": "ALL_OTHERS",
230     "value": "EmptyAddress"
231 }
232 ]

```

Formal Verification Request 177

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

 10, Dec 2019

 1.75 ms

Line 401 in File ERC721BaseToken.sol

```
401 // @CTK NO_BUF_OVERFLOW
```

Line 409-458 in File ERC721BaseToken.sol

```

409 function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory
    data, bool safe) internal {
410     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
411     bool authorized = msg.sender == from ||
412         metaTx ||
413         _superOperators[msg.sender] ||
414         _operatorsForAll[from][msg.sender];
415
416     require(from != address(0), "from is zero address");
417     require(to != address(0), "can't send to zero address");
418
419     uint256 numTokens = ids.length;
420     /*@CTK "_batchTransferFrom_loop"
421     @pre from != address(0)
422     @pre to != address(0)
423     @pre numTokens < 5
424     @inv this._numNFTPerAddress[from] == this._pre._numNFTPerAddress[from]
425     @inv this._numNFTPerAddress[to] == this._pre._numNFTPerAddress[to]
426     @inv ids == ids__pre
427     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from
428         )
429     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> ((msg.sender == from) || (this.
430         _metaTransactionContracts[msg.sender]) || (this._superOperators[msg.sender]) ||
431         (this._operatorsForAll[from][msg.sender])) || (((this._owners[ids[j]] /
432         2**255) == 1) && (this._operators[ids[j]] == msg.sender))
433     @inv i <= numTokens
434     @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
435     @inv numTokens == numTokens__pre
436     @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre.
437         _numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
438     @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
439     @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]

```

```

435     @post i == numTokens
436     @post !__should_return
437     */
438     for(uint256 i = 0; i < numTokens; i++) {
439         uint256 id = ids[i];
440         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
441         require(owner == from, "not owner in batchTransferFrom");
442         require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not
            authorized");
443         _owners[id] = uint256(to);
444         // emit Transfer(from, to, id);
445     }
446     if (from != to) {
447         _numNFTPerAddress[from] -= numTokens;
448         _numNFTPerAddress[to] += numTokens;
449     }
450     if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to,
        ERC721_MANDATORY_RECEIVER))) {
451         require(
452             _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
453             "erc721 batch transfer rejected by to"
454         );
455     }
456 }

```

✓ The code meets the specification.

Formal Verification Request 178

Method will not encounter an assertion failure.



10, Dec 2019



0.92 ms

Line 402 in File ERC721BaseToken.sol

```
402     // @CTK NO_ASF
```

Line 409-458 in File ERC721BaseToken.sol

```

409     function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory
        data, bool safe) internal {
410         bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
411         bool authorized = msg.sender == from ||
412             metaTx ||
413             _superOperators[msg.sender] ||
414             _operatorsForAll[from][msg.sender];
415
416         require(from != address(0), "from is zero address");
417         require(to != address(0), "can't send to zero address");
418
419         uint256 numTokens = ids.length;
420         /* @CTK "_batchTransferFrom_loop"
421             @pre from != address(0)
422             @pre to != address(0)
423             @pre numTokens < 5
424             @inv this._numNFTPerAddress[from] == this._pre._numNFTPerAddress[from]
425             @inv this._numNFTPerAddress[to] == this._pre._numNFTPerAddress[to]

```

```

426     @inv ids == ids__pre
427     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from
428         )
429     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> ((msg.sender == from) || (this.
430         _metaTransactionContracts[msg.sender]) || (this._superOperators[msg.sender]) ||
431         (this._operatorsForAll[from][msg.sender])) || (((this._owners[ids[j]] /
432         2**255) == 1) && (this._operators[ids[j]] == msg.sender))
433     @inv i <= numTokens
434     @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
435     @inv numTokens == numTokens__pre
436     @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre.
437         _numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
438     @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
439     @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
440     @post i == numTokens
441     @post !__should_return
442     */
443     for(uint256 i = 0; i < numTokens; i++) {
444         uint256 id = ids[i];
445         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
446         require(owner == from, "not owner in batchTransferFrom");
447         require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not
448             authorized");
449         _owners[id] = uint256(to);
450         // emit Transfer(from, to, id);
451     }
452     if (from != to) {
453         _numNFTPerAddress[from] -= numTokens;
454         _numNFTPerAddress[to] += numTokens;
455     }
456     if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to,
457         ERC721_MANDATORY_RECEIVER))) {
458         require(
459             _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
460             "erc721 batch transfer rejected by to"
461         );
462     }
463 }

```

✓ The code meets the specification.

Formal Verification Request 179

`__batchTransferFrom`

📅 10, Dec 2019

🕒 874.17 ms

Line 403-408 in File ERC721BaseToken.sol

```

403     /*@CTK FAIL "__batchTransferFrom"
404     @tag assume_completion
405     @pre from != address(0)
406     @pre to != address(0)
407     @post _numNFTPerAddress[from] + _numNFTPerAddress[to] == (__post._numNFTPerAddress[
408         from] + __post._numNFTPerAddress[to])
409     */

```


Line 409-458 in File ERC721BaseToken.sol

```

409     function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory
        data, bool safe) internal {
410         bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
411         bool authorized = msg.sender == from ||
412             metaTx ||
413             _superOperators[msg.sender] ||
414             _operatorsForAll[from][msg.sender];
415
416         require(from != address(0), "from is zero address");
417         require(to != address(0), "can't send to zero address");
418
419         uint256 numTokens = ids.length;
420         /*CTK "_batchTransferFrom_loop"
421             @pre from != address(0)
422             @pre to != address(0)
423             @pre numTokens < 5
424             @inv this._numNFTPerAddress[from] == this._pre._numNFTPerAddress[from]
425             @inv this._numNFTPerAddress[to] == this._pre._numNFTPerAddress[to]
426             @inv ids == ids__pre
427             @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from
428                 )
429             @pre forall j: uint. (j >= 0 /\ j < numTokens) -> ((msg.sender == from) || (this.
430                 _metaTransactionContracts[msg.sender]) || (this._superOperators[msg.sender]) ||
431                 (this._operatorsForAll[from][msg.sender])) || (((this._owners[ids[j]] /
432                 2**255) == 1) && (this._operators[ids[j]] == msg.sender))
433             @inv i <= numTokens
434             @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
435             @inv numTokens == numTokens__pre
436             @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this._pre.
437                 _numNFTPerAddress[from] + this._pre._numNFTPerAddress[to])
438             @post this._numNFTPerAddress[from] == this._pre._numNFTPerAddress[from]
439             @post this._numNFTPerAddress[to] == this._pre._numNFTPerAddress[to]
440             @post i == numTokens
441             @post !_should_return
442             */
443         for(uint256 i = 0; i < numTokens; i++) {
444             uint256 id = ids[i];
445             (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
446             require(owner == from, "not owner in batchTransferFrom");
447             require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not
448                 authorized");
449             _owners[id] = uint256(to);
450             // emit Transfer(from, to, id);
451         }
452         if (from != to) {
453             _numNFTPerAddress[from] -= numTokens;
454             _numNFTPerAddress[to] += numTokens;
455         }
456         if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to,
457             ERC721_MANDATORY_RECEIVER))) {
458             require(
459                 _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
460                 "erc721 batch transfer rejected by to"
461             );
462         }
463     }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     data = ""
5     from = 128
6     ids = [
7       {
8         "key": 0,
9         "value": 2
10      },
11      {
12        "key": "ALL_OTHERS",
13        "value": 0
14      }
15    ]
16    safe = false
17    to = 2
18  }
19  This = 0
20  Internal = {
21    __has_assertion_failure = false
22    __has_buf_overflow = false
23    __has_overflow = false
24    __has_returned = false
25    __reverted = false
26    msg = {
27      "gas": 0,
28      "sender": 0,
29      "value": 0
30    }
31  }
32  Other = {
33    block = {
34      "number": 0,
35      "timestamp": 0
36    }
37  }
38  Address_Map = [
39    {
40      "key": 0,
41      "value": {
42        "contract_name": "ERC721BaseToken",
43        "balance": 0,
44        "contract": {
45          "_ERC721_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
46          "_ERC721_BATCH_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
47          "ERC165ID": "\u00c1\u00c1\u00c1\u00c1",
48          "ERC721_MANDATORY_RECEIVER": "\u00c1\u00c1\u00c1\u00c1",
49          "_numNFTPerAddress": [
50            {
51              "key": 1,
52              "value": 0
53            },
54            {
55              "key": 2,
56              "value": 0
57            },

```

```

58     {
59         "key": 0,
60         "value": 0
61     },
62     {
63         "key": 32,
64         "value": 0
65     },
66     {
67         "key": 8,
68         "value": 0
69     },
70     {
71         "key": 128,
72         "value": 0
73     },
74     {
75         "key": "ALL_OTHERS",
76         "value": 128
77     }
78 ],
79 "_owners": [
80     {
81         "key": "ALL_OTHERS",
82         "value": 128
83     }
84 ],
85 "_operatorsForAll": [
86     {
87         "key": "ALL_OTHERS",
88         "value": [
89             {
90                 "key": "ALL_OTHERS",
91                 "value": false
92             }
93         ]
94     }
95 ],
96 "_operators": [
97     {
98         "key": 16,
99         "value": 64
100     },
101     {
102         "key": 4,
103         "value": 4
104     },
105     {
106         "key": 2,
107         "value": 64
108     },
109     {
110         "key": 0,
111         "value": 0
112     },
113     {
114         "key": "ALL_OTHERS",
115         "value": 128

```

```

116     }
117   ],
118   "_metaTransactionContracts": [
119     {
120       "key": "ALL_OTHERS",
121       "value": false
122     }
123   ],
124   "_admin": 0,
125   "_superOperators": [
126     {
127       "key": "ALL_OTHERS",
128       "value": false
129     }
130   ]
131 }
132 }
133 },
134 {
135   "key": "ALL_OTHERS",
136   "value": "EmptyAddress"
137 }
138 ]
139

```

140 After Execution:

```

141   Input = {
142     data = ""
143     from = 0
144     ids = [
145       {
146         "key": 0,
147         "value": 2
148       },
149       {
150         "key": "ALL_OTHERS",
151         "value": 0
152       }
153     ]
154     safe = false
155     to = 128
156   }
157   This = 0
158   Internal = {
159     __has_assertion_failure = false
160     __has_buf_overflow = false
161     __has_overflow = true
162     __has_returned = false
163     __reverted = false
164     msg = {
165       "gas": 0,
166       "sender": 0,
167       "value": 0
168     }
169   }
170   Other = {
171     block = {
172       "number": 0,
173       "timestamp": 0

```

```

174     }
175 }
176 Address_Map = [
177 {
178     "key": 0,
179     "value": {
180         "contract_name": "ERC721BaseToken",
181         "balance": 0,
182         "contract": {
183             "_ERC721_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
184             "_ERC721_BATCH_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
185             "ERC165ID": "\u00c1\u00c1\u00c1\u00c1",
186             "ERC721_MANDATORY_RECEIVER": "\u00c1\u00c1\u00c1\u00c1",
187             "_numNFTPerAddress": [
188                 {
189                     "key": 1,
190                     "value": 0
191                 },
192                 {
193                     "key": 2,
194                     "value": 0
195                 },
196                 {
197                     "key": 0,
198                     "value": 192
199                 },
200                 {
201                     "key": 32,
202                     "value": 0
203                 },
204                 {
205                     "key": 8,
206                     "value": 0
207                 },
208                 {
209                     "key": 128,
210                     "value": 64
211                 },
212                 {
213                     "key": "ALL_OTHERS",
214                     "value": 128
215                 }
216             ],
217             "_owners": [
218                 {
219                     "key": "ALL_OTHERS",
220                     "value": 128
221                 }
222             ],
223             "_operatorsForAll": [
224                 {
225                     "key": "ALL_OTHERS",
226                     "value": [
227                         {
228                             "key": "ALL_OTHERS",
229                             "value": false
230                         }
231                     ]

```

```

232     }
233   ],
234   "_operators": [
235     {
236       "key": 16,
237       "value": 64
238     },
239     {
240       "key": 4,
241       "value": 4
242     },
243     {
244       "key": 2,
245       "value": 64
246     },
247     {
248       "key": 0,
249       "value": 0
250     },
251     {
252       "key": "ALL_OTHERS",
253       "value": 128
254     }
255   ],
256   "_metaTransactionContracts": [
257     {
258       "key": "ALL_OTHERS",
259       "value": false
260     }
261   ],
262   "_admin": 0,
263   "_superOperators": [
264     {
265       "key": "ALL_OTHERS",
266       "value": false
267     }
268   ]
269 }
270 }
271 },
272 {
273   "key": "ALL_OTHERS",
274   "value": "EmptyAddress"
275 }
276 ]

```

Formal Verification Request 180

supportsInterface



10, Dec 2019



6.47 ms

Line 478-481 in File ERC721BaseToken.sol

```

478   /*@CTK supportsInterface
479     @tag assume_completion

```

```
480     @post __return == (id == 0x01ffc9a7) || (id == 0x80ac58cd)
481     */
```

Line 482-484 in File ERC721BaseToken.sol

```
482     function supportsInterface(bytes4 id) external pure returns (bool) {
483         return id == 0x01ffc9a7 || id == 0x80ac58cd;
484     }
```

✓ The code meets the specification.

Formal Verification Request 181

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 57.41 ms

Line 492 in File ERC721BaseToken.sol

```
492     //@CTK NO_OVERFLOW
```

Line 508-522 in File ERC721BaseToken.sol

```
508     function setApprovalForAllFor(
509         address sender,
510         address operator,
511         bool approved
512     ) external {
513         require(sender != address(0), "Invalid sender address");
514         require(
515             msg.sender == sender ||
516             _metaTransactionContracts[msg.sender] ||
517             _superOperators[msg.sender],
518             "not authorized to approve for all"
519         );
520
521         _setApprovalForAll(sender, operator, approved);
522     }
```

✓ The code meets the specification.

Formal Verification Request 182

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

📅 10, Dec 2019

🕒 5.13 ms

Line 493 in File ERC721BaseToken.sol

```
493     //@CTK NO_BUF_OVERFLOW
```

Line 508-522 in File ERC721BaseToken.sol

```

508     function setApprovalForAllFor(
509         address sender,
510         address operator,
511         bool approved
512     ) external {
513         require(sender != address(0), "Invalid sender address");
514         require(
515             msg.sender == sender ||
516             _metaTransactionContracts[msg.sender] ||
517             _superOperators[msg.sender],
518             "not authorized to approve for all"
519         );
520
521         _setApprovalForAll(sender, operator, approved);
522     }

```

✓ The code meets the specification.

Formal Verification Request 183

Method will not encounter an assertion failure.



10, Dec 2019



5.64 ms

Line 494 in File ERC721BaseToken.sol

```
494 // @CTK NO_ASF
```

Line 508-522 in File ERC721BaseToken.sol

```

508     function setApprovalForAllFor(
509         address sender,
510         address operator,
511         bool approved
512     ) external {
513         require(sender != address(0), "Invalid sender address");
514         require(
515             msg.sender == sender ||
516             _metaTransactionContracts[msg.sender] ||
517             _superOperators[msg.sender],
518             "not authorized to approve for all"
519         );
520
521         _setApprovalForAll(sender, operator, approved);
522     }

```

✓ The code meets the specification.

Formal Verification Request 184

setApprovalForAll require



10, Dec 2019



6.41 ms

Line 495-500 in File ERC721BaseToken.sol


```

495  /*@CTK _setApprovalForAll_require
496  @tag assume_completion
497  @post sender != address(0)
498  @post msg.sender == sender /\ _metaTransactionContracts[msg.sender] == true /\
    _superOperators[msg.sender] == true
499  @post _superOperators[operator] == false
500  */

```

Line 508-522 in File ERC721BaseToken.sol

```

508  function setApprovalForAllFor(
509      address sender,
510      address operator,
511      bool approved
512  ) external {
513      require(sender != address(0), "Invalid sender address");
514      require(
515          msg.sender == sender ||
516          _metaTransactionContracts[msg.sender] ||
517          _superOperators[msg.sender],
518          "not authorized to approve for all"
519      );
520
521      _setApprovalForAll(sender, operator, approved);
522  }

```

✓ The code meets the specification.

Formal Verification Request 185

_setApprovalForAll_change



10, Dec 2019



2.54 ms

Line 501-507 in File ERC721BaseToken.sol

```

501  /*@CTK _setApprovalForAll_change
502  @tag assume_completion
503  @pre sender != address(0)
504  @pre msg.sender == sender /\ _metaTransactionContracts[msg.sender] == true /\
    _superOperators[msg.sender] == true
505  @pre _superOperators[operator] == false
506  @post __post._operatorsForAll[sender][operator] == approved
507  */

```

Line 508-522 in File ERC721BaseToken.sol

```

508  function setApprovalForAllFor(
509      address sender,
510      address operator,
511      bool approved
512  ) external {
513      require(sender != address(0), "Invalid sender address");
514      require(
515          msg.sender == sender ||
516          _metaTransactionContracts[msg.sender] ||
517          _superOperators[msg.sender],

```

```
518         "not authorized to approve for all"
519     );
520
521     _setApprovalForAll(sender, operator, approved);
522 }
```

✓ The code meets the specification.

Formal Verification Request 186

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 21.79 ms

Line 529 in File ERC721BaseToken.sol

```
529 // @CTK_NO_OVERFLOW
```

Line 541-543 in File ERC721BaseToken.sol

```
541 function setApprovalForAll(address operator, bool approved) external {
542     _setApprovalForAll(msg.sender, operator, approved);
543 }
```

✓ The code meets the specification.

Formal Verification Request 187

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

📅 10, Dec 2019

🕒 0.41 ms

Line 530 in File ERC721BaseToken.sol

```
530 // @CTK_NO_BUF_OVERFLOW
```

Line 541-543 in File ERC721BaseToken.sol

```
541 function setApprovalForAll(address operator, bool approved) external {
542     _setApprovalForAll(msg.sender, operator, approved);
543 }
```

✓ The code meets the specification.

Formal Verification Request 188

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.41 ms

Line 531 in File ERC721BaseToken.sol

```
531 // @CTK_NO_ASF
```

Line 541-543 in File ERC721BaseToken.sol

```
541     function setApprovalForAll(address operator, bool approved) external {
542         _setApprovalForAll(msg.sender, operator, approved);
543     }
```

✓ The code meets the specification.

Formal Verification Request 189

setApprovalForAll_require



10, Dec 2019



0.89 ms

Line 532-535 in File ERC721BaseToken.sol

```
532     /*@CTK setApprovalForAll_require
533         @tag assume_completion
534         @post _superOperators[operator] == false
535     */
```

Line 541-543 in File ERC721BaseToken.sol

```
541     function setApprovalForAll(address operator, bool approved) external {
542         _setApprovalForAll(msg.sender, operator, approved);
543     }
```

✓ The code meets the specification.

Formal Verification Request 190

setApprovalForAll_change



10, Dec 2019



1.94 ms

Line 536-540 in File ERC721BaseToken.sol

```
536     /*@CTK setApprovalForAll_change
537         @tag assume_completion
538         @pre _superOperators[operator] == false
539         @post __post._operatorsForAll[msg.sender][operator] == approved
540     */
```

Line 541-543 in File ERC721BaseToken.sol

```
541     function setApprovalForAll(address operator, bool approved) external {
542         _setApprovalForAll(msg.sender, operator, approved);
543     }
```

✓ The code meets the specification.

Formal Verification Request 191

If method completes, integer overflow would not happen.



10, Dec 2019



0.34 ms

Line 545 in File ERC721BaseToken.sol

```
545 // @CTK_NO_OVERFLOW
```

Line 557-569 in File ERC721BaseToken.sol

```
557 function _setApprovalForAll(  
558     address sender,  
559     address operator,  
560     bool approved  
561 ) internal {  
562     require(  
563         !_superOperators[operator],  
564         "super operator can't have their approvalForAll changed"  
565     );  
566     _operatorsForAll[sender][operator] = approved;  
567     emit ApprovalForAll(sender, operator, approved);  
568 }  
569
```

✓ The code meets the specification.

Formal Verification Request 192

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



10, Dec 2019



0.33 ms

Line 546 in File ERC721BaseToken.sol

```
546 // @CTK_NO_BUF_OVERFLOW
```

Line 557-569 in File ERC721BaseToken.sol

```
557 function _setApprovalForAll(  
558     address sender,  
559     address operator,  
560     bool approved  
561 ) internal {  
562     require(  
563         !_superOperators[operator],  
564         "super operator can't have their approvalForAll changed"  
565     );  
566     _operatorsForAll[sender][operator] = approved;  
567     emit ApprovalForAll(sender, operator, approved);  
568 }  
569
```

✓ The code meets the specification.

Formal Verification Request 193

Method will not encounter an assertion failure.



10, Dec 2019



0.33 ms

Line 547 in File ERC721BaseToken.sol

```
547 // @CTK NO_ASF
```

Line 557-569 in File ERC721BaseToken.sol

```
557 function _setApprovalForAll(  
558     address sender,  
559     address operator,  
560     bool approved  
561 ) internal {  
562     require(  
563         !_superOperators[operator],  
564         "super operator can't have their approvalForAll changed"  
565     );  
566     _operatorsForAll[sender][operator] = approved;  
567     emit ApprovalForAll(sender, operator, approved);  
568 }  
569 }
```

✓ The code meets the specification.

Formal Verification Request 194

`__setApprovalForAll__require`



10, Dec 2019



0.71 ms

Line 548-551 in File ERC721BaseToken.sol

```
548 /* @CTK __setApprovalForAll__require  
549     @tag assume_completion  
550     @post _superOperators[operator] == false  
551 */
```

Line 557-569 in File ERC721BaseToken.sol

```
557 function _setApprovalForAll(  
558     address sender,  
559     address operator,  
560     bool approved  
561 ) internal {  
562     require(  
563         !_superOperators[operator],  
564         "super operator can't have their approvalForAll changed"  
565     );  
566     _operatorsForAll[sender][operator] = approved;  
567     emit ApprovalForAll(sender, operator, approved);  
568 }  
569 }
```

✓ The code meets the specification.

Formal Verification Request 195

`__setApprovalForAll__change`



10, Dec 2019



1.67 ms

Line 552-556 in File ERC721BaseToken.sol

```
552  /*@CTK __setApprovalForAll__change
553      @tag assume_completion
554      @pre _superOperators[operator] == false
555      @post __post._operatorsForAll[sender][operator] == approved
556  */
```

Line 557-569 in File ERC721BaseToken.sol

```
557  function __setApprovalForAll(
558      address sender,
559      address operator,
560      bool approved
561  ) internal {
562      require(
563          !_superOperators[operator],
564          "super operator can't have their approvalForAll changed"
565      );
566      _operatorsForAll[sender][operator] = approved;
567
568      emit ApprovalForAll(sender, operator, approved);
569  }
```

✓ The code meets the specification.

Formal Verification Request 196

If method completes, integer overflow would not happen.



10, Dec 2019



5.41 ms

Line 577 in File ERC721BaseToken.sol

```
577  //@CTK NO_OVERFLOW
```

Line 584-590 in File ERC721BaseToken.sol


```
584  function isApprovedForAll(address owner, address operator)
585      external
586      view
587      returns (bool)
588  {
589      return _operatorsForAll[owner][operator] || _superOperators[operator];
590  }
```

✓ The code meets the specification.

Formal Verification Request 197

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

 10, Dec 2019

 0.28 ms

Line 578 in File ERC721BaseToken.sol

578 `//@CTK NO_BUF_OVERFLOW`

Line 584-590 in File ERC721BaseToken.sol


```
584     function isApprovedForAll(address owner, address operator)
585         external
586         view
587         returns (bool)
588     {
589         return _operatorsForAll[owner][operator] || _superOperators[operator];
590     }
```

 The code meets the specification.

Formal Verification Request 198

Method will not encounter an assertion failure.

 10, Dec 2019

 0.29 ms

Line 579 in File ERC721BaseToken.sol

579 `//@CTK NO_ASF`

Line 584-590 in File ERC721BaseToken.sol


```
584     function isApprovedForAll(address owner, address operator)
585         external
586         view
587         returns (bool)
588     {
589         return _operatorsForAll[owner][operator] || _superOperators[operator];
590     }
```

 The code meets the specification.

Formal Verification Request 199

isApprovedForAll

 10, Dec 2019

 1.15 ms

Line 580-583 in File ERC721BaseToken.sol

```

580  /*@CTK isApprovedForAll
581      @post (_operatorsForAll[owner][operator] == true \/ _superOperators[operator] == true)
          -> __return == true
582      @post (_operatorsForAll[owner][operator] == false /\ _superOperators[operator] ==
          false) -> __return == false
583  */

```

Line 584-590 in File ERC721BaseToken.sol

```

584  function isApprovedForAll(address owner, address operator)
585      external
586      view
587      returns (bool)
588  {
589      return _operatorsForAll[owner][operator] || _superOperators[operator];
590  }

```

✓ The code meets the specification.

Formal Verification Request 200

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 39.11 ms

Line 592 in File ERC721BaseToken.sol

```

592  //@CTK FAIL NO_OVERFLOW

```

Line 605-610 in File ERC721BaseToken.sol

```

605  function _burn(address from, address owner, uint256 id) public {
606      require(from == owner, "not owner");
607      _owners[id] = 2**160; // cannot mint it again
608      _numNFTPerAddress[from]--;
609      emit Transfer(from, address(0), id);
610  }

```

✗ This code violates the specification.

```

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

```



```

19     }
20 }
21 Other = {
22     block = {
23         "number": 0,
24         "timestamp": 0
25     }
26 }
27 Address_Map = [
28     {
29         "key": 0,
30         "value": {
31             "contract_name": "ERC721BaseToken",
32             "balance": 0,
33             "contract": {
34                 "_ERC721_RECEIVED": "AAAA",
35                 "_ERC721_BATCH_RECEIVED": "AAAA",
36                 "ERC165ID": "AAAA",
37                 "ERC721_MANDATORY_RECEIVER": "AAAA",
38                 "_numNFTPerAddress": [
39                     {
40                         "key": 0,
41                         "value": 0
42                     },
43                     {
44                         "key": 68,
45                         "value": 16
46                     },
47                     {
48                         "key": 64,
49                         "value": 0
50                     },
51                     {
52                         "key": 32,
53                         "value": 0
54                     },
55                     {
56                         "key": 1,
57                         "value": 0
58                     },
59                     {
60                         "key": 16,
61                         "value": 0
62                     },
63                     {
64                         "key": 4,
65                         "value": 8
66                     },
67                     {
68                         "key": 128,
69                         "value": 8
70                     },
71                     {
72                         "key": 12,
73                         "value": 0
74                     },
75                     {
76                         "key": "ALL_OTHERS",

```

```

77         "value": 255
78     }
79 ],
80 "_owners": [
81     {
82         "key": 8,
83         "value": 16
84     },
85     {
86         "key": 0,
87         "value": 2
88     },
89     {
90         "key": 32,
91         "value": 2
92     },
93     {
94         "key": 64,
95         "value": 64
96     },
97     {
98         "key": 1,
99         "value": 1
100    },
101    {
102        "key": "ALL_OTHERS",
103        "value": 0
104    }
105 ],
106 "_operatorsForAll": [
107     {
108         "key": "ALL_OTHERS",
109         "value": [
110             {
111                 "key": "ALL_OTHERS",
112                 "value": false
113             }
114         ]
115     }
116 ],
117 "_operators": [
118     {
119         "key": 0,
120         "value": 8
121     },
122     {
123         "key": 32,
124         "value": 128
125     },
126     {
127         "key": "ALL_OTHERS",
128         "value": 0
129     }
130 ],
131 "_metaTransactionContracts": [
132     {
133         "key": 0,
134         "value": true

```

```

135         },
136         {
137             "key": "ALL_OTHERS",
138             "value": false
139         }
140     ],
141     "_admin": 0,
142     "_superOperators": [
143         {
144             "key": "ALL_OTHERS",
145             "value": false
146         }
147     ]
148 }
149 }
150 },
151 {
152     "key": "ALL_OTHERS",
153     "value": "EmptyAddress"
154 }
155 ]
156
157 After Execution:
158 Input = {
159     from = 0
160     id = 0
161     owner = 0
162 }
163 This = 0
164 Internal = {
165     __has_assertion_failure = false
166     __has_buf_overflow = false
167     __has_overflow = true
168     __has_returned = false
169     __reverted = false
170     msg = {
171         "gas": 0,
172         "sender": 0,
173         "value": 0
174     }
175 }
176 Other = {
177     block = {
178         "number": 0,
179         "timestamp": 0
180     }
181 }
182 Address_Map = [
183     {
184         "key": 0,
185         "value": {
186             "contract_name": "ERC721BaseToken",
187             "balance": 0,
188             "contract": {
189                 "_ERC721_RECEIVED": "AAAA",
190                 "_ERC721_BATCH_RECEIVED": "AAAA",
191                 "ERC165ID": "AAAA",
192                 "ERC721_MANDATORY_RECEIVER": "AAAA",

```

```

193     "_numNFTPerAddress": [
194         {
195             "key": 68,
196             "value": 16
197         },
198         {
199             "key": 64,
200             "value": 0
201         },
202         {
203             "key": 32,
204             "value": 0
205         },
206         {
207             "key": 1,
208             "value": 0
209         },
210         {
211             "key": 128,
212             "value": 8
213         },
214         {
215             "key": 16,
216             "value": 0
217         },
218         {
219             "key": 4,
220             "value": 8
221         },
222         {
223             "key": 12,
224             "value": 0
225         },
226         {
227             "key": "ALL_OTHERS",
228             "value": 255
229         }
230     ],
231     "_owners": [
232         {
233             "key": 8,
234             "value": 16
235         },
236         {
237             "key": 32,
238             "value": 2
239         },
240         {
241             "key": 64,
242             "value": 64
243         },
244         {
245             "key": 1,
246             "value": 1
247         },
248         {
249             "key": "ALL_OTHERS",
250             "value": 0

```

```

251     }
252   ],
253   "_operatorsForAll": [
254     {
255       "key": "ALL_OTHERS",
256       "value": [
257         {
258           "key": "ALL_OTHERS",
259           "value": false
260         }
261       ]
262     }
263   ],
264   "_operators": [
265     {
266       "key": 0,
267       "value": 8
268     },
269     {
270       "key": 32,
271       "value": 128
272     },
273     {
274       "key": "ALL_OTHERS",
275       "value": 0
276     }
277   ],
278   "_metaTransactionContracts": [
279     {
280       "key": 0,
281       "value": true
282     },
283     {
284       "key": "ALL_OTHERS",
285       "value": false
286     }
287   ],
288   "_admin": 0,
289   "_superOperators": [
290     {
291       "key": "ALL_OTHERS",
292       "value": false
293     }
294   ]
295 }
296 }
297 },
298 {
299   "key": "ALL_OTHERS",
300   "value": "EmptyAddress"
301 }
302 ]

```

Formal Verification Request 201

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

📅 10, Dec 2019

🕒 0.69 ms

Line 593 in File ERC721BaseToken.sol

```
593 // @CTK_NO_BUF_OVERFLOW
```

Line 605-610 in File ERC721BaseToken.sol

```
605 function _burn(address from, address owner, uint256 id) public {
606     require(from == owner, "not owner");
607     _owners[id] = 2**160; // cannot mint it again
608     _numNFTPerAddress[from]--;
609     emit Transfer(from, address(0), id);
610 }
```

✅ The code meets the specification.

Formal Verification Request 202

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.88 ms

Line 594 in File ERC721BaseToken.sol

```
594 // @CTK_NO_ASF
```

Line 605-610 in File ERC721BaseToken.sol

```
605 function _burn(address from, address owner, uint256 id) public {
606     require(from == owner, "not owner");
607     _owners[id] = 2**160; // cannot mint it again
608     _numNFTPerAddress[from]--;
609     emit Transfer(from, address(0), id);
610 }
```

✅ The code meets the specification.

Formal Verification Request 203

__burn__require

📅 10, Dec 2019

🕒 0.46 ms

Line 595-598 in File ERC721BaseToken.sol

```
595 /* @CTK __burn__require
596     @tag assume_completion
597     @post from == owner
598 */
```

Line 605-610 in File ERC721BaseToken.sol

```
605     function _burn(address from, address owner, uint256 id) public {
606         require(from == owner, "not owner");
607         _owners[id] = 2**160; // cannot mint it again
608         _numNFTPerAddress[from]--;
609         emit Transfer(from, address(0), id);
610     }
```

✓ The code meets the specification.

Formal Verification Request 204

__burn__change

📅 10, Dec 2019

🕒 2.11 ms

Line 599-604 in File ERC721BaseToken.sol

```
599     /*@CTK __burn__change
600         @tag assume_completion
601         @pre from == owner
602         @post __post._owners[id] == 2**160
603         @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
604     */
```

Line 605-610 in File ERC721BaseToken.sol

```
605     function _burn(address from, address owner, uint256 id) public {
606         require(from == owner, "not owner");
607         _owners[id] = 2**160; // cannot mint it again
608         _numNFTPerAddress[from]--;
609         emit Transfer(from, address(0), id);
610     }
```

✓ The code meets the specification.

Formal Verification Request 205

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 56.73 ms

Line 614 in File ERC721BaseToken.sol

```
614     /*@CTK FAIL NO_OVERFLOW
```

Line 627-629 in File ERC721BaseToken.sol

```
627     function burn(uint256 id) external {
628         _burn(msg.sender, _ownerOf(id), id);
629     }
```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     id = 0
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    block = {
21      "number": 0,
22      "timestamp": 0
23    }
24  }
25  Address_Map = [
26    {
27      "key": 0,
28      "value": {
29        "contract_name": "ERC721BaseToken",
30        "balance": 0,
31        "contract": {
32          "_ERC721_RECEIVED": "AAAA",
33          "_ERC721_BATCH_RECEIVED": "AAAA",
34          "ERC165ID": "AAAA",
35          "ERC721_MANDATORY_RECEIVER": "AAAA",
36          "_numNFTPerAddress": [
37            {
38              "key": 32,
39              "value": 1
40            },
41            {
42              "key": 16,
43              "value": 16
44            },
45            {
46              "key": 8,
47              "value": 8
48            },
49            {
50              "key": "ALL_OTHERS",
51              "value": 0
52            }
53          ],
54          "_owners": [
55            {
56              "key": 64,
57              "value": 32
58            },

```



```

59     {
60         "key": "ALL_OTHERS",
61         "value": 0
62     }
63 ],
64 "_operatorsForAll": [
65     {
66         "key": "ALL_OTHERS",
67         "value": [
68             {
69                 "key": "ALL_OTHERS",
70                 "value": false
71             }
72         ]
73     }
74 ],
75 "_operators": [
76     {
77         "key": 0,
78         "value": 64
79     },
80     {
81         "key": 33,
82         "value": 32
83     },
84     {
85         "key": "ALL_OTHERS",
86         "value": 0
87     }
88 ],
89 "_metaTransactionContracts": [
90     {
91         "key": 0,
92         "value": true
93     },
94     {
95         "key": "ALL_OTHERS",
96         "value": false
97     }
98 ],
99 "_admin": 0,
100 "_superOperators": [
101     {
102         "key": "ALL_OTHERS",
103         "value": false
104     }
105 ]
106 }
107 }
108 },
109 {
110     "key": "ALL_OTHERS",
111     "value": "EmptyAddress"
112 }
113 ]
114
115 After Execution:
116 Input = {

```

```

117     id = 0
118 }
119 This = 0
120 Internal = {
121     __has_assertion_failure = false
122     __has_buf_overflow = false
123     __has_overflow = true
124     __has_returned = false
125     __reverted = false
126     msg = {
127         "gas": 0,
128         "sender": 0,
129         "value": 0
130     }
131 }
132 Other = {
133     block = {
134         "number": 0,
135         "timestamp": 0
136     }
137 }
138 Address_Map = [
139     {
140         "key": 0,
141         "value": {
142             "contract_name": "ERC721BaseToken",
143             "balance": 0,
144             "contract": {
145                 "_ERC721_RECEIVED": "AAAA",
146                 "_ERC721_BATCH_RECEIVED": "AAAA",
147                 "ERC165ID": "AAAA",
148                 "ERC721_MANDATORY_RECEIVER": "AAAA",
149                 "_numNFTPerAddress": [
150                     {
151                         "key": 32,
152                         "value": 1
153                     },
154                     {
155                         "key": 0,
156                         "value": 255
157                     },
158                     {
159                         "key": 16,
160                         "value": 16
161                     },
162                     {
163                         "key": 8,
164                         "value": 8
165                     },
166                     {
167                         "key": "ALL_OTHERS",
168                         "value": 0
169                     }
170                 ],
171                 "_owners": [
172                     {
173                         "key": 64,
174                         "value": 32

```

```

175     },
176     {
177         "key": "ALL_OTHERS",
178         "value": 0
179     }
180 ],
181 "_operatorsForAll": [
182     {
183         "key": "ALL_OTHERS",
184         "value": [
185             {
186                 "key": "ALL_OTHERS",
187                 "value": false
188             }
189         ]
190     }
191 ],
192 "_operators": [
193     {
194         "key": 0,
195         "value": 64
196     },
197     {
198         "key": 33,
199         "value": 32
200     },
201     {
202         "key": "ALL_OTHERS",
203         "value": 0
204     }
205 ],
206 "_metaTransactionContracts": [
207     {
208         "key": 0,
209         "value": true
210     },
211     {
212         "key": "ALL_OTHERS",
213         "value": false
214     }
215 ],
216 "_admin": 0,
217 "_superOperators": [
218     {
219         "key": "ALL_OTHERS",
220         "value": false
221     }
222 ]
223 }
224 }
225 },
226 {
227     "key": "ALL_OTHERS",
228     "value": "EmptyAddress"
229 }
230 ]

```

Formal Verification Request 206

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

📅 10, Dec 2019

🕒 0.66 ms

Line 615 in File ERC721BaseToken.sol

```
615 // @CTK NO_BUF_OVERFLOW
```

Line 627-629 in File ERC721BaseToken.sol

```
627 function burn(uint256 id) external {  
628     _burn(msg.sender, _ownerOf(id), id);  
629 }
```

✅ The code meets the specification.

Formal Verification Request 207

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.55 ms

Line 616 in File ERC721BaseToken.sol

```
616 // @CTK NO_ASF
```

Line 627-629 in File ERC721BaseToken.sol

```
627 function burn(uint256 id) external {  
628     _burn(msg.sender, _ownerOf(id), id);  
629 }
```

✅ The code meets the specification.

Formal Verification Request 208

burn_require

📅 10, Dec 2019

🕒 1.05 ms

Line 617-620 in File ERC721BaseToken.sol

```
617 /* @CTK burn_require  
618    @tag assume_completion  
619    @post msg.sender == address(_owners[id])  
620    */
```

Line 627-629 in File ERC721BaseToken.sol

```
627 function burn(uint256 id) external {  
628     _burn(msg.sender, _ownerOf(id), id);  
629 }
```

✅ The code meets the specification.

Formal Verification Request 209

burn_change



10, Dec 2019



5.75 ms

Line 621-626 in File ERC721BaseToken.sol

```
621  /*@CTK burn_change
622    @tag assume_completion
623    @pre msg.sender == address(_owners[id])
624    @post __post._owners[id] == 2**160
625    @post __post._numNFTPerAddress[msg.sender] == _numNFTPerAddress[msg.sender] - 1
626  */
```

Line 627-629 in File ERC721BaseToken.sol

```
627  function burn(uint256 id) external {
628    _burn(msg.sender, _ownerOf(id), id);
629  }
```

✓ The code meets the specification.

Formal Verification Request 210

If method completes, integer overflow would not happen.



10, Dec 2019



62.08 ms

Line 634 in File ERC721BaseToken.sol

```
634  //@CTK NO_OVERFLOW
```

Line 650-662 in File ERC721BaseToken.sol

```
650  function burnFrom(address from, uint256 id) external {
651    require(from != address(0), "Invalid sender address");
652    (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
653    require(
654      msg.sender == from ||
655      _metaTransactionContracts[msg.sender] ||
656      (operatorEnabled && _operators[id] == msg.sender) ||
657      _superOperators[msg.sender] ||
658      _operatorsForAll[from][msg.sender],
659      "not authorized to burn"
660    );
661    _burn(from, owner, id);
662  }
```

✓ The code meets the specification.

Formal Verification Request 211

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



10, Dec 2019



8.24 ms

Line 635 in File ERC721BaseToken.sol

```
635 // @CTK NO_BUF_OVERFLOW
```

Line 650-662 in File ERC721BaseToken.sol

```
650 function burnFrom(address from, uint256 id) external {
651     require(from != address(0), "Invalid sender address");
652     (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
653     require(
654         msg.sender == from ||
655         _metaTransactionContracts[msg.sender] ||
656         (operatorEnabled && _operators[id] == msg.sender) ||
657         _superOperators[msg.sender] ||
658         _operatorsForAll[from][msg.sender],
659         "not authorized to burn"
660     );
661     _burn(from, owner, id);
662 }
```

✓ The code meets the specification.

Formal Verification Request 212

burnFrom_require



10, Dec 2019



12.49 ms

Line 636-641 in File ERC721BaseToken.sol

```
636 /* @CTK burnFrom_require
637    @tag assume_completion
638    @post from != address(0)
639    @post (msg.sender == from) || _metaTransactionContracts[msg.sender] || ((_owners[id] /
        2**255) == 1 && _operators[id] == msg.sender) || _superOperators[msg.sender] ||
        _operatorsForAll[from][msg.sender]
640    @post from == address(_owners[id])
641    */
```

Line 650-662 in File ERC721BaseToken.sol

```
650 function burnFrom(address from, uint256 id) external {
651     require(from != address(0), "Invalid sender address");
652     (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
653     require(
654         msg.sender == from ||
655         _metaTransactionContracts[msg.sender] ||
656         (operatorEnabled && _operators[id] == msg.sender) ||
657         _superOperators[msg.sender] ||
658         _operatorsForAll[from][msg.sender],
659         "not authorized to burn"
660     );
661     _burn(from, owner, id);
662 }
```

✓ The code meets the specification.

Formal Verification Request 213

burnFrom_change

📅 10, Dec 2019

🕒 3.37 ms

Line 642-649 in File ERC721BaseToken.sol

```

642  /*@CTK burnFrom_change
643  @tag assume_completion
644  @pre from != address(0)
645  @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || ((_owners[id] /
      2**255) == 1 && _operators[id] == msg.sender) || _superOperators[msg.sender] ||
      _operatorsForAll[from][msg.sender]
646  @pre from == address(_owners[id])
647  @post __post._owners[id] == 2**160
648  @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
649  */

```

Line 650-662 in File ERC721BaseToken.sol

```

650  function burnFrom(address from, uint256 id) external {
651  require(from != address(0), "Invalid sender address");
652  (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
653  require(
654  msg.sender == from ||
655  _metaTransactionContracts[msg.sender] ||
656  (operatorEnabled && _operators[id] == msg.sender) ||
657  _superOperators[msg.sender] ||
658  _operatorsForAll[from][msg.sender],
659  "not authorized to burn"
660  );
661  _burn(from, owner, id);
662  }

```

✅ The code meets the specification.

Formal Verification Request 214

__batchTransferFrom_loop__Generated

📅 10, Dec 2019

🕒 95.37 ms

(Loop) Line 420-437 in File ERC721BaseToken.sol

```

420  /*@CTK "__batchTransferFrom_loop"
421  @pre from != address(0)
422  @pre to != address(0)
423  @pre numTokens < 5
424  @inv this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
425  @inv this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
426  @inv ids == ids__pre
427  @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from
      )

```

```

428     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> ((msg.sender == from) || (this.
        _metaTransactionContracts[msg.sender]) || (this._superOperators[msg.sender]) ||
        (this._operatorsForAll[from][msg.sender])) || (((this._owners[ids[j]] /
        2**255) == 1) && (this._operators[ids[j]] == msg.sender))
429     @inv i <= numTokens
430     @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
431     @inv numTokens == numTokens__pre
432     @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre.
        _numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
433     @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
434     @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
435     @post i == numTokens
436     @post !__should_return
437     */

```

(Loop) Line 420-445 in File ERC721BaseToken.sol

```

420     /*@CTK "_batchTransferFrom_loop"
421     @pre from != address(0)
422     @pre to != address(0)
423     @pre numTokens < 5
424     @inv this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
425     @inv this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
426     @inv ids == ids__pre
427     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from
        )
428     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> ((msg.sender == from) || (this.
        _metaTransactionContracts[msg.sender]) || (this._superOperators[msg.sender]) ||
        (this._operatorsForAll[from][msg.sender])) || (((this._owners[ids[j]] /
        2**255) == 1) && (this._operators[ids[j]] == msg.sender))
429     @inv i <= numTokens
430     @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
431     @inv numTokens == numTokens__pre
432     @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre.
        _numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
433     @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
434     @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
435     @post i == numTokens
436     @post !__should_return
437     */
438     for(uint256 i = 0; i < numTokens; i++) {
439         uint256 id = ids[i];
440         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
441         require(owner == from, "not owner in batchTransferFrom");
442         require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not
            authorized");
443         _owners[id] = uint256(to);
444         // emit Transfer(from, to, id);
445     }

```

✓ The code meets the specification.

Formal Verification Request 215

If method completes, integer overflow would not happen.



10, Dec 2019



13.45 ms

Line 14 in File SuperOperators.sol

```
14 // @CTK_NO_OVERFLOW
```

Line 26-33 in File SuperOperators.sol

```
26 function setSuperOperator(address superOperator, bool enabled) external {
27     require(
28         msg.sender == _admin,
29         "only admin is allowed to add super operators"
30     );
31     _superOperators[superOperator] = enabled;
32     emit SuperOperator(superOperator, enabled);
33 }
```

✓ The code meets the specification.

Formal Verification Request 216

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



10, Dec 2019



0.52 ms

Line 15 in File SuperOperators.sol

```
15 // @CTK_NO_BUF_OVERFLOW
```

Line 26-33 in File SuperOperators.sol

```
26 function setSuperOperator(address superOperator, bool enabled) external {
27     require(
28         msg.sender == _admin,
29         "only admin is allowed to add super operators"
30     );
31     _superOperators[superOperator] = enabled;
32     emit SuperOperator(superOperator, enabled);
33 }
```

✓ The code meets the specification.

Formal Verification Request 217

Method will not encounter an assertion failure.



10, Dec 2019



0.33 ms

Line 16 in File SuperOperators.sol

```
16 // @CTK_NO_ASF
```

Line 26-33 in File SuperOperators.sol

```
26 function setSuperOperator(address superOperator, bool enabled) external {
27     require(
28         msg.sender == _admin,
```

```

29         "only admin is allowed to add super operators"
30     );
31     _superOperators[superOperator] = enabled;
32     emit SuperOperator(superOperator, enabled);
33 }


```

✓ The code meets the specification.

Formal Verification Request 218

setSuperOperator_admin

 10, Dec 2019

 0.25 ms

Line 17-20 in File SuperOperators.sol

```

17     /*@CTK setSuperOperator_admin
18         @tag assume_completion
19         @inv msg.sender == _admin
20     */

```

Line 26-33 in File SuperOperators.sol

```

26     function setSuperOperator(address superOperator, bool enabled) external {
27         require(
28             msg.sender == _admin,
29             "only admin is allowed to add super operators"
30         );
31         _superOperators[superOperator] = enabled;
32         emit SuperOperator(superOperator, enabled);
33     }


```

✓ The code meets the specification.

Formal Verification Request 219

setSuperOperator_change

 10, Dec 2019

 1.45 ms

Line 21-25 in File SuperOperators.sol

```

21     /*@CTK setSuperOperator_change
22         @tag assume_completion
23         @pre msg.sender == _admin
24         @post __post._superOperators[superOperator] == enabled
25     */

```

Line 26-33 in File SuperOperators.sol

```

26     function setSuperOperator(address superOperator, bool enabled) external {
27         require(
28             msg.sender == _admin,
29             "only admin is allowed to add super operators"
30         );

```

```
31     _superOperators[superOperator] = enabled;
32     emit SuperOperator(superOperator, enabled);
33 }
```

✓ The code meets the specification.

Formal Verification Request 220

If method completes, integer overflow would not happen.



10, Dec 2019



4.32 ms

Line 38 in File SuperOperators.sol

```
38 //@CTK NO_OVERFLOW
```

Line 45-47 in File SuperOperators.sol

```
45     function isSuperOperator(address who) public view returns (bool) {
46         return _superOperators[who];
47     }
```

✓ The code meets the specification.

Formal Verification Request 221

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



10, Dec 2019



0.26 ms

Line 39 in File SuperOperators.sol

```
39 //@CTK NO_BUF_OVERFLOW
```

Line 45-47 in File SuperOperators.sol

```
45     function isSuperOperator(address who) public view returns (bool) {
46         return _superOperators[who];
47     }
```

✓ The code meets the specification.

Formal Verification Request 222

Method will not encounter an assertion failure.



10, Dec 2019



0.26 ms

Line 40 in File SuperOperators.sol

```
40 //@CTK NO_ASF
```

Line 45-47 in File SuperOperators.sol

```

45     function isSuperOperator(address who) public view returns (bool) {
46         return _superOperators[who];
47     }

```

✓ The code meets the specification.

Formal Verification Request 223

isSuperOperator



10, Dec 2019



0.28 ms

Line 41-44 in File SuperOperators.sol

```

41     /*@CTK isSuperOperator
42         @tag assume_completion
43         @post __return == _superOperators[who]
44     */

```

Line 45-47 in File SuperOperators.sol

```

45     function isSuperOperator(address who) public view returns (bool) {
46         return _superOperators[who];
47     }

```

✓ The code meets the specification.

Formal Verification Request 224

If method completes, integer overflow would not happen.



10, Dec 2019



3.46 ms

Line 5 in File AddressUtils.sol

```

5     //@CTK NO_OVERFLOW

```

Line 8-10 in File AddressUtils.sol

```

8     function toPayable(address _address) internal pure returns (address payable _payable) {
9         return address(uint160(_address));
10    }

```

✓ The code meets the specification.

Formal Verification Request 225

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



10, Dec 2019



0.26 ms

Line 6 in File AddressUtils.sol

```
6 //@CTK NO_BUF_OVERFLOW
```

Line 8-10 in File AddressUtils.sol

```
8 function toPayable(address _address) internal pure returns (address payable _payable) {  
9     return address(uint160(_address));  
10 }
```

✓ The code meets the specification.

Formal Verification Request 226

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.25 ms

Line 7 in File AddressUtils.sol

```
7 //@CTK NO_ASF
```

Line 8-10 in File AddressUtils.sol

```
8 function toPayable(address _address) internal pure returns (address payable _payable) {  
9     return address(uint160(_address));  
10 }
```

✓ The code meets the specification.

Formal Verification Request 227

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 108.0 ms

Line 8 in File Land.sol

```
8 //@CTK NO_OVERFLOW
```

Line 16-23 in File Land.sol

```
16 constructor(  
17     address metaTransactionContract,  
18     address admin  
19 ) public LandBaseToken(  
20     metaTransactionContract,  
21     admin  
22 ) {  
23 }
```

✓ The code meets the specification.

Formal Verification Request 228

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



10, Dec 2019



0.56 ms

Line 9 in File Land.sol

```
9 // @CTK_NO_BUF_OVERFLOW
```

Line 16-23 in File Land.sol

```
16 constructor(  
17     address metaTransactionContract,  
18     address admin  
19 ) public LandBaseToken(  
20     metaTransactionContract,  
21     admin  
22 ) {  
23 }
```

✓ The code meets the specification.

Formal Verification Request 229

Method will not encounter an assertion failure.



10, Dec 2019



0.56 ms

Line 10 in File Land.sol

```
10 // @CTK_NO_ASF
```

Line 16-23 in File Land.sol

```
16 constructor(  
17     address metaTransactionContract,  
18     address admin  
19 ) public LandBaseToken(  
20     metaTransactionContract,  
21     admin  
22 ) {  
23 }
```

✓ The code meets the specification.

Formal Verification Request 230

Land



10, Dec 2019



2.14 ms

Line 11-15 in File Land.sol

```
11  /*@CTK Land
12     @tag assume_completion
13     @post __post._admin == admin
14     @post __post._metaTransactionContracts[metaTransactionContract] == true
15  */
```

Line 16-23 in File Land.sol

```
16  constructor(
17      address metaTransactionContract,
18      address admin
19  ) public LandBaseToken(
20      metaTransactionContract,
21      admin
22  ) {
23  }
```

✓ The code meets the specification.

Formal Verification Request 231

name



10, Dec 2019



4.9 ms

Line 29-31 in File Land.sol

```
29  /*@CTK name
30     @post __return == "Sandbox's LANDs"
31  */
```

Line 32-34 in File Land.sol

```
32  function name() external pure returns (string memory) {
33      return "Sandbox's LANDs";
34  }
```

✓ The code meets the specification.

Formal Verification Request 232

symbol



10, Dec 2019



4.41 ms

Line 40-42 in File Land.sol

```
40  /*@CTK symbol
41     @post __return == "LAND"
42  */
```

Line 43-45 in File Land.sol

```
43  function symbol() external pure returns (string memory) {
44      return "LAND";
45  }
```

✓ The code meets the specification.

Formal Verification Request 233

If method completes, integer overflow would not happen.

10, Dec 2019

7.37 ms

Line 93 in File Land.sol

```
93 //@CTK NO_OVERFLOW
```

Line 101-103 in File Land.sol

```
101 function supportsInterface(bytes4 id) external pure returns (bool) {
102     return id == 0x01ffc9a7 || id == 0x80ac58cd || id == 0x5b5e139f;
103 }
```

✓ The code meets the specification.

Formal Verification Request 234

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

10, Dec 2019

0.28 ms

Line 94 in File Land.sol

```
94 //@CTK NO_BUF_OVERFLOW
```

Line 101-103 in File Land.sol

```
101 function supportsInterface(bytes4 id) external pure returns (bool) {
102     return id == 0x01ffc9a7 || id == 0x80ac58cd || id == 0x5b5e139f;
103 }
```

✓ The code meets the specification.

Formal Verification Request 235

Method will not encounter an assertion failure.

10, Dec 2019

0.27 ms

Line 95 in File Land.sol

```
95 //@CTK NO_ASF
```

Line 101-103 in File Land.sol

```
101 function supportsInterface(bytes4 id) external pure returns (bool) {
102     return id == 0x01ffc9a7 || id == 0x80ac58cd || id == 0x5b5e139f;
103 }
```

✓ The code meets the specification.

Formal Verification Request 236

supportsInterface



10, Dec 2019



1.81 ms

Line 96-100 in File Land.sol

```
96      /*@CTK supportsInterface
97      @tag assume_completion
98      @post (id == 0x01ffc9a7 \\/ id == 0x80ac58cd \\/ id == 0x5b5e139f) -> __return == true
99      @post (id != 0x01ffc9a7 /\ id != 0x80ac58cd /\ id != 0x5b5e139f) -> __return == false
100     */
```

Line 101-103 in File Land.sol

```
101     function supportsInterface(bytes4 id) external pure returns (bool) {
102         return id == 0x01ffc9a7 || id == 0x80ac58cd || id == 0x5b5e139f;
103     }
```

✓ The code meets the specification.

Formal Verification Request 237

If method completes, integer overflow would not happen.



10, Dec 2019



15.97 ms

Line 23 in File LandBaseToken.sol

```
23      //@CTK NO_OVERFLOW
```

Line 34-41 in File LandBaseToken.sol

```
34     function setMinter(address minter, bool enabled) external {
35         require(
36             msg.sender == _admin,
37             "only admin is allowed to add minters"
38         );
39         _minters[minter] = enabled;
40         emit Minter(minter, enabled);
41     }
```

✓ The code meets the specification.

Formal Verification Request 238

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



10, Dec 2019



0.39 ms

Line 24 in File LandBaseToken.sol

```
24      //@CTK NO_BUF_OVERFLOW
```

Line 34-41 in File LandBaseToken.sol

```

34     function setMinter(address minter, bool enabled) external {
35         require(
36             msg.sender == _admin,
37             "only admin is allowed to add minters"
38         );
39         _minters[minter] = enabled;
40         emit Minter(minter, enabled);
41     }

```

✓ The code meets the specification.

Formal Verification Request 239

Method will not encounter an assertion failure.



10, Dec 2019



0.38 ms

Line 25 in File LandBaseToken.sol

```

25     // @CTK NO_ASF

```

Line 34-41 in File LandBaseToken.sol

```

34     function setMinter(address minter, bool enabled) external {
35         require(
36             msg.sender == _admin,
37             "only admin is allowed to add minters"
38         );
39         _minters[minter] = enabled;
40         emit Minter(minter, enabled);
41     }

```

✓ The code meets the specification.

Formal Verification Request 240

setMinter_require



10, Dec 2019



1.21 ms

Line 26-29 in File LandBaseToken.sol

```

26     /* @CTK setMinter_require
27         @tag assume_completion
28         @post msg.sender == _admin
29     */

```

Line 34-41 in File LandBaseToken.sol

```

34     function setMinter(address minter, bool enabled) external {
35         require(
36             msg.sender == _admin,
37             "only admin is allowed to add minters"

```

```
38     );  
39     _minters[minter] = enabled;  
40     emit Minter(minter, enabled);  
41 }
```

✓ The code meets the specification.

Formal Verification Request 241

setMinter_change



10, Dec 2019



3.9 ms

Line 30-33 in File LandBaseToken.sol

```
30  /*@CTK setMinter_change  
31  @tag assume_completion  
32  @post __post._minters[minter] == enabled  
33  */
```

Line 34-41 in File LandBaseToken.sol

```
34  function setMinter(address minter, bool enabled) external {  
35      require(  
36          msg.sender == _admin,  
37          "only admin is allowed to add minters"  
38      );  
39      _minters[minter] = enabled;  
40      emit Minter(minter, enabled);  
41  }
```

✓ The code meets the specification.

Formal Verification Request 242

If method completes, integer overflow would not happen.



10, Dec 2019



4.51 ms

Line 46 in File LandBaseToken.sol

```
46  //@CTK NO_OVERFLOW
```

Line 53-55 in File LandBaseToken.sol

```
53  function isMinter(address who) public view returns (bool) {  
54      return _minters[who];  
55  }
```

✓ The code meets the specification.

Formal Verification Request 243

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

📅 10, Dec 2019

🕒 0.33 ms

Line 47 in File LandBaseToken.sol

```
47 // @CTK NO_BUF_OVERFLOW
```

Line 53-55 in File LandBaseToken.sol

```
53 function isMinter(address who) public view returns (bool) {  
54     return _minters[who];  
55 }
```

✅ The code meets the specification.

Formal Verification Request 244

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 0.33 ms

Line 48 in File LandBaseToken.sol

```
48 // @CTK NO_ASF
```

Line 53-55 in File LandBaseToken.sol

```
53 function isMinter(address who) public view returns (bool) {  
54     return _minters[who];  
55 }
```

✅ The code meets the specification.

Formal Verification Request 245

isMinter

📅 10, Dec 2019

🕒 0.33 ms

Line 49-52 in File LandBaseToken.sol

```
49 /* @CTK isMinter  
50    @tag assume_completion  
51    @post __return == _minters[who]  
52    */
```

Line 53-55 in File LandBaseToken.sol

```
53 function isMinter(address who) public view returns (bool) {  
54     return _minters[who];  
55 }
```

✅ The code meets the specification.

Formal Verification Request 246

If method completes, integer overflow would not happen.



10, Dec 2019



58.47 ms

Line 57 in File LandBaseToken.sol

```
57 // @CTK_NO_OVERFLOW
```

Line 65-69 in File LandBaseToken.sol

```
65 constructor(  
66     address metaTransactionContract,  
67     address admin  
68 ) public ERC721BaseToken(metaTransactionContract, admin) {  
69 }
```

✓ The code meets the specification.

Formal Verification Request 247

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



10, Dec 2019



0.47 ms

Line 58 in File LandBaseToken.sol

```
58 // @CTK_NO_BUF_OVERFLOW
```

Line 65-69 in File LandBaseToken.sol

```
65 constructor(  
66     address metaTransactionContract,  
67     address admin  
68 ) public ERC721BaseToken(metaTransactionContract, admin) {  
69 }
```

✓ The code meets the specification.

Formal Verification Request 248

Method will not encounter an assertion failure.



10, Dec 2019



0.47 ms

Line 59 in File LandBaseToken.sol

```
59 // @CTK_NO_ASF
```

Line 65-69 in File LandBaseToken.sol

```
65 constructor(  
66     address metaTransactionContract,  
67     address admin  
68 ) public ERC721BaseToken(metaTransactionContract, admin) {  
69 }
```

✓ The code meets the specification.

Formal Verification Request 249

LandBaseToken

10, Dec 2019

1.8 ms

Line 60-64 in File LandBaseToken.sol

```
60  /*@CTK LandBaseToken
61     @tag assume_completion
62     @post __post._admin == admin
63     @post __post._metaTransactionContracts[metaTransactionContract] == true
64  */
```

Line 65-69 in File LandBaseToken.sol

```
65  constructor(
66      address metaTransactionContract,
67      address admin
68  ) public ERC721BaseToken(metaTransactionContract, admin) {
69  }
```

✓ The code meets the specification.

Formal Verification Request 250

width

10, Dec 2019

3.8 ms

Line 73-75 in File LandBaseToken.sol

```
73  /*@CTK width
74     @post __return == GRID_SIZE
75  */
```

Line 76-78 in File LandBaseToken.sol

```
76  function width() external returns(uint256) {
77      return GRID_SIZE;
78  }
```

✓ The code meets the specification.

Formal Verification Request 251

height

10, Dec 2019

3.8 ms

Line 82-84 in File LandBaseToken.sol

```
82  /*@CTK height
83  @post __return == GRID_SIZE
84  */
```

Line 85-87 in File LandBaseToken.sol

```
85  function height() external returns(uint256) {
86      return GRID_SIZE;
87  }
```

✓ The code meets the specification.

Formal Verification Request 252

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 127.07 ms

Line 92 in File LandBaseToken.sol

```
92  //@CTK NO_OVERFLOW
```

Line 101-104 in File LandBaseToken.sol

```
101  function x(uint256 id) external returns(uint256) {
102      require(_ownerOf(id) != address(0), "token does not exist");
103      return id % GRID_SIZE;
104  }
```

✓ The code meets the specification.

Formal Verification Request 253

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

📅 10, Dec 2019

🕒 1.73 ms

Line 93 in File LandBaseToken.sol

```
93  //@CTK NO_BUF_OVERFLOW
```

Line 101-104 in File LandBaseToken.sol


```
101  function x(uint256 id) external returns(uint256) {
102      require(_ownerOf(id) != address(0), "token does not exist");
103      return id % GRID_SIZE;
104  }
```

✓ The code meets the specification.

Formal Verification Request 254

Method will not encounter an assertion failure.

 10, Dec 2019

 1.16 ms

Line 94 in File LandBaseToken.sol

```
94 // @CTK NO_ASF
```

Line 101-104 in File LandBaseToken.sol


```
101 function x(uint256 id) external returns(uint256) {  
102     require(_ownerOf(id) != address(0), "token does not exist");  
103     return id % GRID_SIZE;  
104 }
```

 The code meets the specification.

Formal Verification Request 255

x

 10, Dec 2019

 1.08 ms

Line 95-100 in File LandBaseToken.sol

```
95 /* @CTK x  
96     @tag assume_completion  
97     @pre GRID_SIZE == 408  
98     @pre address(_owners[id]) != address(0)  
99     @post __return == id % GRID_SIZE  
100 */
```

Line 101-104 in File LandBaseToken.sol


```
101 function x(uint256 id) external returns(uint256) {  
102     require(_ownerOf(id) != address(0), "token does not exist");  
103     return id % GRID_SIZE;  
104 }
```

 The code meets the specification.

Formal Verification Request 256

If method completes, integer overflow would not happen.

 10, Dec 2019

 79.7 ms

Line 109 in File LandBaseToken.sol

```
109 // @CTK NO_OVERFLOW
```

Line 118-121 in File LandBaseToken.sol


```
118 function y(uint256 id) external returns(uint256) {
119     require(_ownerOf(id) != address(0), "token does not exist");
120     return id / GRID_SIZE;
121 }
```

✓ The code meets the specification.

Formal Verification Request 257

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



10, Dec 2019



0.99 ms

Line 110 in File LandBaseToken.sol

```
110 //CTK NO_BUF_OVERFLOW
```

Line 118-121 in File LandBaseToken.sol

```
118 function y(uint256 id) external returns(uint256) {
119     require(_ownerOf(id) != address(0), "token does not exist");
120     return id / GRID_SIZE;
121 }
```

✓ The code meets the specification.

Formal Verification Request 258

Method will not encounter an assertion failure.



10, Dec 2019



1.03 ms

Line 111 in File LandBaseToken.sol

```
111 //CTK NO_ASF
```

Line 118-121 in File LandBaseToken.sol

```
118 function y(uint256 id) external returns(uint256) {
119     require(_ownerOf(id) != address(0), "token does not exist");
120     return id / GRID_SIZE;
121 }
```

✓ The code meets the specification.

Formal Verification Request 259

y



10, Dec 2019



1.04 ms

Line 112-117 in File LandBaseToken.sol

```

112  /*@CTK y
113    @tag assume_completion
114    @pre GRID_SIZE == 408
115    @pre address(_owners[id]) != address(0)
116    @post __return == id / GRID_SIZE
117  */

```

Line 118-121 in File LandBaseToken.sol

```

118  function y(uint256 id) external returns(uint256) {
119    require(_ownerOf(id) != address(0), "token does not exist");
120    return id / GRID_SIZE;
121  }


```

✓ The code meets the specification.

Formal Verification Request 260

If method completes, integer overflow would not happen.

 10, Dec 2019

 471.12 ms

Line 131 in File LandBaseToken.sol

```

131  //@CTK FAIL NO_OVERFLOW

```

Line 167-264 in File LandBaseToken.sol

```

167  function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data)
      external {
168    require(to != address(0), "to is zero address");
169    require(
170      isMinter(msg.sender),
171      "Only a minter can mint"
172    );
173    require(x % size == 0 && y % size == 0, "Invalid coordinates");
174    require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
175
176    uint256 quadId;
177    uint256 id = x + y * GRID_SIZE;
178
179    if (size == 1) {
180      quadId = id;
181    } else if (size == 3) {
182      quadId = LAYER_3x3 + id;
183    } else if (size == 6) {
184      quadId = LAYER_6x6 + id;
185    } else if (size == 12) {
186      quadId = LAYER_12x12 + id;
187    } else if (size == 24) {
188      quadId = LAYER_24x24 + id;
189    } else {
190      require(false, "Invalid size");
191    }
192
193    require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
      Already minted as 24x24");

```

```

194
195     uint256 toX = x+size;
196     uint256 toY = y+size;
197     if (size <= 12) {
198         require(
199             _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
200             "Already minted as 12x12"
201         );
202     } else {
203         /*@CTK mintQuad_loop1
204          @tag assume_completion
205          @inv x12i <= x + size
206          @post x12i == x + size
207          */
208         for (uint256 x12i = x; x12i < toX; x12i += 12) {
209             for (uint256 y12i = y; y12i < toY; y12i += 12) {
210                 uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
211                 require(_owners[id12x12] == 0, "Already minted as 12x12");
212             }
213         }
214     }
215
216     if (size <= 6) {
217         require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
218             minted as 6x6");
219     } else {
220         for (uint256 x6i = x; x6i < toX; x6i += 6) {
221             for (uint256 y6i = y; y6i < toY; y6i += 6) {
222                 uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
223                 require(_owners[id6x6] == 0, "Already minted as 6x6");
224             }
225         }
226
227         if (size <= 3) {
228             require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
229                 minted as 3x3");
230         } else {
231             for (uint256 x3i = x; x3i < toX; x3i += 3) {
232                 for (uint256 y3i = y; y3i < toY; y3i += 3) {
233                     uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
234                     require(_owners[id3x3] == 0, "Already minted as 3x3");
235                 }
236             }
237
238             /*@CTK mintQuad_loopx
239              @tag assume_completion
240              @pre GRID_SIZE == 408
241              @inv i <= size * size
242              @post i == size * size
243              */
244             for (uint256 i = 0; i < size*size; i++) {
245                 uint256 id = _idInPath(i, size, x, y);
246                 require(_owners[id] == 0, "Already minted");
247                 emit Transfer(address(0), to, id);
248             }
249

```

```

250     _owners[quadId] = uint256(to);
251     _numNFTPerAddress[to] += size * size;
252
253     _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
254 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     data = ""
5     size = 12
6     to = 128
7     x = 0
8     y = 0
9   }
10  This = 0
11  Internal = {
12    __has_assertion_failure = false
13    __has_buf_overflow = false
14    __has_overflow = false
15    __has_returned = false
16    __reverted = false
17    msg = {
18      "gas": 0,
19      "sender": 0,
20      "value": 0
21    }
22  }
23  Other = {
24    block = {
25      "number": 0,
26      "timestamp": 0
27    }
28  }
29  Address_Map = [
30    {
31      "key": 0,
32      "value": {
33        "contract_name": "LandBaseToken",
34        "balance": 0,
35        "contract": {
36          "GRID_SIZE": 64,
37          "LAYER": 0,
38          "LAYER_1x1": 32,
39          "LAYER_3x3": 0,
40          "LAYER_6x6": 2,
41          "LAYER_12x12": 0,
42          "LAYER_24x24": 0,
43          "_minters": [
44            {
45              "key": 8,
46              "value": true
47            },
48            {
49              "key": 0,
50              "value": true
51            },

```

```

52     {
53         "key": "ALL_OTHERS",
54         "value": false
55     }
56 ],
57 "_ERC721_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
58 "_ERC721_BATCH_RECEIVED": "GGGG",
59 "ERC165ID": "\u00c1\u00c1\u00c1\u00c1",
60 "ERC721_MANDATORY_RECEIVER": "GGGG",
61 "_numNFTPerAddress": [
62     {
63         "key": 0,
64         "value": 4
65     },
66     {
67         "key": "ALL_OTHERS",
68         "value": 128
69     }
70 ],
71 "_owners": [
72     {
73         "key": 1,
74         "value": 0
75     },
76     {
77         "key": 80,
78         "value": 0
79     },
80     {
81         "key": 0,
82         "value": 0
83     },
84     {
85         "key": 128,
86         "value": 0
87     },
88     {
89         "key": "ALL_OTHERS",
90         "value": 128
91     }
92 ],
93 "_operatorsForAll": [
94     {
95         "key": "ALL_OTHERS",
96         "value": [
97             {
98                 "key": "ALL_OTHERS",
99                 "value": false
100             }
101         ]
102     }
103 ],
104 "_operators": [
105     {
106         "key": 1,
107         "value": 0
108     },
109     {

```

```

110         "key": 80,
111         "value": 0
112     },
113     {
114         "key": 0,
115         "value": 0
116     },
117     {
118         "key": 128,
119         "value": 0
120     },
121     {
122         "key": "ALL_OTHERS",
123         "value": 128
124     }
125 ],
126 "_metaTransactionContracts": [
127     {
128         "key": 0,
129         "value": true
130     },
131     {
132         "key": "ALL_OTHERS",
133         "value": false
134     }
135 ],
136 "_admin": 0,
137 "_superOperators": [
138     {
139         "key": 2,
140         "value": true
141     },
142     {
143         "key": "ALL_OTHERS",
144         "value": false
145     }
146 ]
147 }
148 }
149 },
150 {
151     "key": "ALL_OTHERS",
152     "value": "EmptyAddress"
153 }
154 ]

```

156 After Execution:

```

157     Input = {
158         data = ""
159         size = 12
160         to = 128
161         x = 0
162         y = 0
163     }
164     This = 0
165     Internal = {
166         __has_assertion_failure = false
167         __has_buf_overflow = false

```

```

168     __has_overflow = true
169     __has_returned = false
170     __reverted = false
171     msg = {
172         "gas": 0,
173         "sender": 0,
174         "value": 0
175     }
176 }
177 Other = {
178     block = {
179         "number": 0,
180         "timestamp": 0
181     }
182 }
183 Address_Map = [
184     {
185         "key": 0,
186         "value": {
187             "contract_name": "LandBaseToken",
188             "balance": 0,
189             "contract": {
190                 "GRID_SIZE": 64,
191                 "LAYER": 0,
192                 "LAYER_1x1": 32,
193                 "LAYER_3x3": 0,
194                 "LAYER_6x6": 2,
195                 "LAYER_12x12": 0,
196                 "LAYER_24x24": 0,
197                 "_minters": [
198                     {
199                         "key": 8,
200                         "value": true
201                     },
202                     {
203                         "key": 0,
204                         "value": true
205                     },
206                     {
207                         "key": "ALL_OTHERS",
208                         "value": false
209                     }
210                 ],
211                 "_ERC721_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
212                 "_ERC721_BATCH_RECEIVED": "GGGG",
213                 "ERC165ID": "\u00c1\u00c1\u00c1\u00c1",
214                 "ERC721_MANDATORY_RECEIVER": "GGGG",
215                 "_numNFTPerAddress": [
216                     {
217                         "key": 0,
218                         "value": 4
219                     },
220                     {
221                         "key": 128,
222                         "value": 16
223                     },
224                     {
225                         "key": "ALL_OTHERS",

```

```

226         "value": 128
227     }
228 ],
229 "_owners": [
230     {
231         "key": 1,
232         "value": 0
233     },
234     {
235         "key": 80,
236         "value": 0
237     },
238     {
239         "key": 128,
240         "value": 0
241     },
242     {
243         "key": "ALL_OTHERS",
244         "value": 128
245     }
246 ],
247 "_operatorsForAll": [
248     {
249         "key": "ALL_OTHERS",
250         "value": [
251             {
252                 "key": "ALL_OTHERS",
253                 "value": false
254             }
255         ]
256     }
257 ],
258 "_operators": [
259     {
260         "key": 1,
261         "value": 0
262     },
263     {
264         "key": 80,
265         "value": 0
266     },
267     {
268         "key": 0,
269         "value": 0
270     },
271     {
272         "key": 128,
273         "value": 0
274     },
275     {
276         "key": "ALL_OTHERS",
277         "value": 128
278     }
279 ],
280 "_metaTransactionContracts": [
281     {
282         "key": 0,
283         "value": true

```



```

284         },
285         {
286             "key": "ALL_OTHERS",
287             "value": false
288         }
289     ],
290     "_admin": 0,
291     "_superOperators": [
292         {
293             "key": 2,
294             "value": true
295         },
296         {
297             "key": "ALL_OTHERS",
298             "value": false
299         }
300     ]
301 }
302 }
303 },
304 {
305     "key": "ALL_OTHERS",
306     "value": "EmptyAddress"
307 }
308 ]

```

Formal Verification Request 261

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



10, Dec 2019



58.84 ms

Line 132 in File LandBaseToken.sol

```
132 // @CTK NO_BUF_OVERFLOW
```

Line 167-264 in File LandBaseToken.sol

```

167 function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data)
168     external {
169     require(to != address(0), "to is zero address");
170     require(
171         isMinter(msg.sender),
172         "Only a minter can mint"
173     );
174     require(x % size == 0 && y % size == 0, "Invalid coordinates");
175     require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
176
177     uint256 quadId;
178     uint256 id = x + y * GRID_SIZE;
179
180     if (size == 1) {
181         quadId = id;
182     } else if (size == 3) {
183         quadId = LAYER_3x3 + id;
184     } else if (size == 6) {
185         quadId = LAYER_6x6 + id;

```

```

185     } else if (size == 12) {
186         quadId = LAYER_12x12 + id;
187     } else if (size == 24) {
188         quadId = LAYER_24x24 + id;
189     } else {
190         require(false, "Invalid size");
191     }
192
193     require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
        Already minted as 24x24");
194
195     uint256 toX = x+size;
196     uint256 toY = y+size;
197     if (size <= 12) {
198         require(
199             _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
200             "Already minted as 12x12"
201         );
202     } else {
203         /*@CTK mintQuad_loop1
204          @tag assume_completion
205          @inv x12i <= x + size
206          @post x12i == x + size
207          */
208         for (uint256 x12i = x; x12i < toX; x12i += 12) {
209             for (uint256 y12i = y; y12i < toY; y12i += 12) {
210                 uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
211                 require(_owners[id12x12] == 0, "Already minted as 12x12");
212             }
213         }
214     }
215
216     if (size <= 6) {
217         require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
            minted as 6x6");
218     } else {
219         for (uint256 x6i = x; x6i < toX; x6i += 6) {
220             for (uint256 y6i = y; y6i < toY; y6i += 6) {
221                 uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
222                 require(_owners[id6x6] == 0, "Already minted as 6x6");
223             }
224         }
225     }
226
227     if (size <= 3) {
228         require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
            minted as 3x3");
229     } else {
230         for (uint256 x3i = x; x3i < toX; x3i += 3) {
231             for (uint256 y3i = y; y3i < toY; y3i += 3) {
232                 uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
233                 require(_owners[id3x3] == 0, "Already minted as 3x3");
234             }
235         }
236     }
237
238     /*@CTK mintQuad_loopx
239     @tag assume_completion

```

```

240     @pre GRID_SIZE == 408
241     @inv i <= size * size
242     @post i == size * size
243     */
244     for (uint256 i = 0; i < size*size; i++) {
245         uint256 id = _idInPath(i, size, x, y);
246         require(_owners[id] == 0, "Already minted");
247         emit Transfer(address(0), to, id);
248     }
249
250     _owners[quadId] = uint256(to);
251     _numNFTPerAddress[to] += size * size;
252
253     _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
254 }

```

✓ The code meets the specification.

Formal Verification Request 262

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 67.33 ms

Line 133 in File LandBaseToken.sol

```
133 // @CTK NO_ASF
```

Line 167-264 in File LandBaseToken.sol

```

167     function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data)
168         external {
169         require(to != address(0), "to is zero address");
170         require(
171             isMinter(msg.sender),
172             "Only a minter can mint"
173         );
174         require(x % size == 0 && y % size == 0, "Invalid coordinates");
175         require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
176
177         uint256 quadId;
178         uint256 id = x + y * GRID_SIZE;
179
180         if (size == 1) {
181             quadId = id;
182         } else if (size == 3) {
183             quadId = LAYER_3x3 + id;
184         } else if (size == 6) {
185             quadId = LAYER_6x6 + id;
186         } else if (size == 12) {
187             quadId = LAYER_12x12 + id;
188         } else if (size == 24) {
189             quadId = LAYER_24x24 + id;
190         } else {
191             require(false, "Invalid size");
192         }

```

```

193     require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
        Already minted as 24x24");
194
195     uint256 toX = x+size;
196     uint256 toY = y+size;
197     if (size <= 12) {
198         require(
199             _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
200             "Already minted as 12x12"
201         );
202     } else {
203         /*@CTK mintQuad_loop1
204            @tag assume_completion
205            @inv x12i <= x + size
206            @post x12i == x + size
207            */
208         for (uint256 x12i = x; x12i < toX; x12i += 12) {
209             for (uint256 y12i = y; y12i < toY; y12i += 12) {
210                 uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
211                 require(_owners[id12x12] == 0, "Already minted as 12x12");
212             }
213         }
214     }
215
216     if (size <= 6) {
217         require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
            minted as 6x6");
218     } else {
219         for (uint256 x6i = x; x6i < toX; x6i += 6) {
220             for (uint256 y6i = y; y6i < toY; y6i += 6) {
221                 uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
222                 require(_owners[id6x6] == 0, "Already minted as 6x6");
223             }
224         }
225     }
226
227     if (size <= 3) {
228         require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
            minted as 3x3");
229     } else {
230         for (uint256 x3i = x; x3i < toX; x3i += 3) {
231             for (uint256 y3i = y; y3i < toY; y3i += 3) {
232                 uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
233                 require(_owners[id3x3] == 0, "Already minted as 3x3");
234             }
235         }
236     }
237
238     /*@CTK mintQuad_loopx
239        @tag assume_completion
240        @pre GRID_SIZE == 408
241        @inv i <= size * size
242        @post i == size * size
243        */
244     for (uint256 i = 0; i < size*size; i++) {
245         uint256 id = _idInPath(i, size, x, y);
246         require(_owners[id] == 0, "Already minted");
247         emit Transfer(address(0), to, id);

```

```

248     }
249
250     _owners[quadId] = uint256(to);
251     _numNFTPerAddress[to] += size * size;
252
253     _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
254 }

```

✓ The code meets the specification.

Formal Verification Request 263

mintQuad_require

📅 10, Dec 2019

🕒 2672.73 ms

Line 134-146 in File LandBaseToken.sol

```

134  /*@CTK mintQuad_require
135   @tag assume_completion
136   @pre GRID_SIZE == 408
137   @post to != address(0)
138   @post _minters[msg.sender] == true
139   @post (x % size == 0) /\ (y % size == 0)
140   @post (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
141   @post (size == 1 /\ size == 3 /\ size == 6 /\ size == 12 /\ size == 24)
142   @post _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0
143   @post size <= 12 -> _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] ==
      0
144   @post size <= 6 -> _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0
145   @post size <= 3 -> _owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0
146   */

```

Line 167-264 in File LandBaseToken.sol

```

167  function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data)
      external {
168      require(to != address(0), "to is zero address");
169      require(
170          isMinter(msg.sender),
171          "Only a minter can mint"
172      );
173      require(x % size == 0 && y % size == 0, "Invalid coordinates");
174      require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
175
176      uint256 quadId;
177      uint256 id = x + y * GRID_SIZE;
178
179      if (size == 1) {
180          quadId = id;
181      } else if (size == 3) {
182          quadId = LAYER_3x3 + id;
183      } else if (size == 6) {
184          quadId = LAYER_6x6 + id;
185      } else if (size == 12) {
186          quadId = LAYER_12x12 + id;
187      } else if (size == 24) {

```

```

188     quadId = LAYER_24x24 + id;
189   } else {
190     require(false, "Invalid size");
191   }
192
193   require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
    Already minted as 24x24");
194
195   uint256 toX = x+size;
196   uint256 toY = y+size;
197   if (size <= 12) {
198     require(
199       _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
200       "Already minted as 12x12"
201     );
202   } else {
203     /*@CTK mintQuad_loop1
204      @tag assume_completion
205      @inv x12i <= x + size
206      @post x12i == x + size
207      */
208     for (uint256 x12i = x; x12i < toX; x12i += 12) {
209       for (uint256 y12i = y; y12i < toY; y12i += 12) {
210         uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
211         require(_owners[id12x12] == 0, "Already minted as 12x12");
212       }
213     }
214   }
215
216   if (size <= 6) {
217     require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
    minted as 6x6");
218   } else {
219     for (uint256 x6i = x; x6i < toX; x6i += 6) {
220       for (uint256 y6i = y; y6i < toY; y6i += 6) {
221         uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
222         require(_owners[id6x6] == 0, "Already minted as 6x6");
223       }
224     }
225   }
226
227   if (size <= 3) {
228     require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
    minted as 3x3");
229   } else {
230     for (uint256 x3i = x; x3i < toX; x3i += 3) {
231       for (uint256 y3i = y; y3i < toY; y3i += 3) {
232         uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
233         require(_owners[id3x3] == 0, "Already minted as 3x3");
234       }
235     }
236   }
237
238   /*@CTK mintQuad_loopx
239   @tag assume_completion
240   @pre GRID_SIZE == 408
241   @inv i <= size * size
242   @post i == size * size

```

```

243     */
244     for (uint256 i = 0; i < size*size; i++) {
245         uint256 id = _idInPath(i, size, x, y);
246         require(_owners[id] == 0, "Already minted");
247         emit Transfer(address(0), to, id);
248     }
249
250     _owners[quadId] = uint256(to);
251     _numNFTPerAddress[to] += size * size;
252
253     _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
254 }

```

✔ The code meets the specification.

Formal Verification Request 264

mintQuad_change

 10, Dec 2019

 231.83 ms

Line 147-166 in File LandBaseToken.sol

[illegible]

Line 167-264 in File LandBaseToken.sol

```

167     function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data)
168         external {
169             require(to != address(0), "to is zero address");
170             require(
171                 isMinter(msg.sender),
172                 "Only a minter can mint"
173             );
174             require(x % size == 0 && y % size == 0, "Invalid coordinates");
175             require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
176
177             uint256 quadId;

```

```

177     uint256 id = x + y * GRID_SIZE;
178
179     if (size == 1) {
180         quadId = id;
181     } else if (size == 3) {
182         quadId = LAYER_3x3 + id;
183     } else if (size == 6) {
184         quadId = LAYER_6x6 + id;
185     } else if (size == 12) {
186         quadId = LAYER_12x12 + id;
187     } else if (size == 24) {
188         quadId = LAYER_24x24 + id;
189     } else {
190         require(false, "Invalid size");
191     }
192
193     require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
        Already minted as 24x24");
194
195     uint256 toX = x+size;
196     uint256 toY = y+size;
197     if (size <= 12) {
198         require(
199             _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
200             "Already minted as 12x12"
201         );
202     } else {
203         /*@CTK mintQuad_loop1
204          @tag assume_completion
205          @inv x12i <= x + size
206          @post x12i == x + size
207          */
208         for (uint256 x12i = x; x12i < toX; x12i += 12) {
209             for (uint256 y12i = y; y12i < toY; y12i += 12) {
210                 uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
211                 require(_owners[id12x12] == 0, "Already minted as 12x12");
212             }
213         }
214     }
215
216     if (size <= 6) {
217         require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
            minted as 6x6");
218     } else {
219         for (uint256 x6i = x; x6i < toX; x6i += 6) {
220             for (uint256 y6i = y; y6i < toY; y6i += 6) {
221                 uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
222                 require(_owners[id6x6] == 0, "Already minted as 6x6");
223             }
224         }
225     }
226
227     if (size <= 3) {
228         require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
            minted as 3x3");
229     } else {
230         for (uint256 x3i = x; x3i < toX; x3i += 3) {
231             for (uint256 y3i = y; y3i < toY; y3i += 3) {

```



```

232         uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
233         require(_owners[id3x3] == 0, "Already minted as 3x3");
234     }
235 }
236 }
237
238 /*@CTK mintQuad_loopx
239   @tag assume_completion
240   @pre GRID_SIZE == 408
241   @inv i <= size * size
242   @post i == size * size
243 */
244 for (uint256 i = 0; i < size*size; i++) {
245     uint256 id = _idInPath(i, size, x, y);
246     require(_owners[id] == 0, "Already minted");
247     emit Transfer(address(0), to, id);
248 }
249
250 _owners[quadId] = uint256(to);
251 _numNFTPerAddress[to] += size * size;
252
253 _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
254 }

```

✓ The code meets the specification.

Formal Verification Request 265

If method completes, integer overflow would not happen.



10, Dec 2019



36.46 ms

Line 266 in File LandBaseToken.sol

```
266 // @CTK FAIL NO_OVERFLOW
```

Line 275-282 in File LandBaseToken.sol

```

275 function _idInPath(uint256 i, uint256 size, uint256 x, uint256 y) internal pure returns(
276     uint256) {
277     uint256 row = i / size;
278     if(row % 2 == 0) { // allow ids to follow a path in a quad
279         return (x + (i%size)) + ((y + row) * GRID_SIZE);
280     } else {
281         return ((x + size) - (1 + i%size)) + ((y + row) * GRID_SIZE);
282     }
283 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     i = 32
5     size = 130
6     x = 32
7     y = 50

```

```

8   }
9   This = 0
10  Internal = {
11    __has_assertion_failure = false
12    __has_buf_overflow = false
13    __has_overflow = false
14    __has_returned = false
15    __reverted = false
16    msg = {
17      "gas": 0,
18      "sender": 0,
19      "value": 0
20    }
21  }
22  Other = {
23    __return = 0
24    block = {
25      "number": 0,
26      "timestamp": 0
27    }
28  }
29  Address_Map = [
30    {
31      "key": "ALL_OTHERS",
32      "value": {
33        "contract_name": "LandBaseToken",
34        "balance": 0,
35        "contract": {
36          "GRID_SIZE": 18,
37          "LAYER": 0,
38          "LAYER_1x1": 0,
39          "LAYER_3x3": 0,
40          "LAYER_6x6": 0,
41          "LAYER_12x12": 0,
42          "LAYER_24x24": 0,
43          "_minters": [
44            {
45              "key": 0,
46              "value": false
47            },
48            {
49              "key": "ALL_OTHERS",
50              "value": true
51            }
52          ],
53          "_ERC721_RECEIVED": "AAAA",
54          "_ERC721_BATCH_RECEIVED": "AAAA",
55          "ERC165ID": "AAAA",
56          "ERC721_MANDATORY_RECEIVER": "AAAA",
57          "_numNFTPerAddress": [
58            {
59              "key": 64,
60              "value": 2
61            },
62            {
63              "key": 0,
64              "value": 0
65            }

```

```

66     {
67         "key": 4,
68         "value": 2
69     },
70     {
71         "key": "ALL_OTHERS",
72         "value": 16
73     }
74 ],
75 "_owners": [
76     {
77         "key": 0,
78         "value": 1
79     },
80     {
81         "key": 8,
82         "value": 128
83     },
84     {
85         "key": 2,
86         "value": 32
87     },
88     {
89         "key": "ALL_OTHERS",
90         "value": 0
91     }
92 ],
93 "_operatorsForAll": [
94     {
95         "key": "ALL_OTHERS",
96         "value": [
97             {
98                 "key": 0,
99                 "value": false
100             },
101             {
102                 "key": "ALL_OTHERS",
103                 "value": true
104             }
105         ]
106     }
107 ],
108 "_operators": [
109     {
110         "key": 32,
111         "value": 16
112     },
113     {
114         "key": 0,
115         "value": 8
116     },
117     {
118         "key": "ALL_OTHERS",
119         "value": 0
120     }
121 ],
122 "_metaTransactionContracts": [
123     {

```

```

124         "key": "ALL_OTHERS",
125         "value": true
126     }
127 ],
128     "_admin": 0,
129     "_superOperators": [
130     {
131         "key": "ALL_OTHERS",
132         "value": false
133     }
134 ]
135 }
136 }
137 }
138 ]
139
140 After Execution:
141 Input = {
142     i = 32
143     size = 130
144     x = 32
145     y = 50
146 }
147 This = 0
148 Internal = {
149     __has_assertion_failure = false
150     __has_buf_overflow = false
151     __has_overflow = true
152     __has_returned = true
153     __reverted = false
154     msg = {
155         "gas": 0,
156         "sender": 0,
157         "value": 0
158     }
159 }
160 Other = {
161     __return = 196
162     block = {
163         "number": 0,
164         "timestamp": 0
165     }
166 }
167 Address_Map = [
168 {
169     "key": "ALL_OTHERS",
170     "value": {
171         "contract_name": "LandBaseToken",
172         "balance": 0,
173         "contract": {
174             "GRID_SIZE": 18,
175             "LAYER": 0,
176             "LAYER_1x1": 0,
177             "LAYER_3x3": 0,
178             "LAYER_6x6": 0,
179             "LAYER_12x12": 0,
180             "LAYER_24x24": 0,
181             "_minters": [

```

```

182     {
183         "key": 0,
184         "value": false
185     },
186     {
187         "key": "ALL_OTHERS",
188         "value": true
189     }
190 ],
191 "_ERC721_RECEIVED": "AAAA",
192 "_ERC721_BATCH_RECEIVED": "AAAA",
193 "ERC165ID": "AAAA",
194 "ERC721_MANDATORY_RECEIVER": "AAAA",
195 "_numNFTPerAddress": [
196     {
197         "key": 64,
198         "value": 2
199     },
200     {
201         "key": 0,
202         "value": 0
203     },
204     {
205         "key": 4,
206         "value": 2
207     },
208     {
209         "key": "ALL_OTHERS",
210         "value": 16
211     }
212 ],
213 "_owners": [
214     {
215         "key": 0,
216         "value": 1
217     },
218     {
219         "key": 8,
220         "value": 128
221     },
222     {
223         "key": 2,
224         "value": 32
225     },
226     {
227         "key": "ALL_OTHERS",
228         "value": 0
229     }
230 ],
231 "_operatorsForAll": [
232     {
233         "key": "ALL_OTHERS",
234         "value": [
235             {
236                 "key": 0,
237                 "value": false
238             },
239             {

```

```

240         "key": "ALL_OTHERS",
241         "value": true
242     }
243 ]
244 }
245 ],
246 "_operators": [
247     {
248         "key": 32,
249         "value": 16
250     },
251     {
252         "key": 0,
253         "value": 8
254     },
255     {
256         "key": "ALL_OTHERS",
257         "value": 0
258     }
259 ],
260 "_metaTransactionContracts": [
261     {
262         "key": "ALL_OTHERS",
263         "value": true
264     }
265 ],
266 "_admin": 0,
267 "_superOperators": [
268     {
269         "key": "ALL_OTHERS",
270         "value": false
271     }
272 ]
273 }
274 }
275 }
276 ]

```

Formal Verification Request 266

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



10, Dec 2019



0.51 ms

Line 267 in File LandBaseToken.sol

```
267 // @CTK NO_BUF_OVERFLOW
```

Line 275-282 in File LandBaseToken.sol

```

275 function _idInPath(uint256 i, uint256 size, uint256 x, uint256 y) internal pure returns(
276     uint256) {
277     uint256 row = i / size;
278     if(row % 2 == 0) { // allow ids to follow a path in a quad
279         return (x + (i%size)) + ((y + row) * GRID_SIZE);
280     } else {
281         return ((x + size) - (1 + i%size)) + ((y + row) * GRID_SIZE);

```

```

281     }
282 }

```

✓ The code meets the specification.

Formal Verification Request 267

Method will not encounter an assertion failure.



10, Dec 2019



10.38 ms

Line 268 in File LandBaseToken.sol

```

268 //CTK FAIL NO_ASF

```

Line 275-282 in File LandBaseToken.sol

```

275 function _idInPath(uint256 i, uint256 size, uint256 x, uint256 y) internal pure returns(
    uint256) {
276     uint256 row = i / size;
277     if(row % 2 == 0) { // allow ids to follow a path in a quad
278         return (x + (i%size)) + ((y + row) * GRID_SIZE);
279     } else {
280         return ((x + size) - (1 + i%size)) + ((y + row) * GRID_SIZE);
281     }
282 }

```

✗ This code violates the specification.

```

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

```

```

29 Address_Map = [
30     {
31         "key": "ALL_OTHERS",
32         "value": {
33             "contract_name": "LandBaseToken",
34             "balance": 0,
35             "contract": {
36                 "GRID_SIZE": 0,
37                 "LAYER": 0,
38                 "LAYER_1x1": 0,
39                 "LAYER_3x3": 0,
40                 "LAYER_6x6": 0,
41                 "LAYER_12x12": 0,
42                 "LAYER_24x24": 0,
43                 "_minters": [
44                     {
45                         "key": 0,
46                         "value": true
47                     },
48                     {
49                         "key": "ALL_OTHERS",
50                         "value": false
51                     }
52                 ],
53                 "_ERC721_RECEIVED": "AAAA",
54                 "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
55                 "ERC165ID": "AAAA",
56                 "ERC721_MANDATORY_RECEIVER": "AAAA",
57                 "_numNFTPerAddress": [
58                     {
59                         "key": 64,
60                         "value": 32
61                     },
62                     {
63                         "key": 32,
64                         "value": 64
65                     },
66                     {
67                         "key": 0,
68                         "value": 128
69                     },
70                     {
71                         "key": 2,
72                         "value": 0
73                     },
74                     {
75                         "key": "ALL_OTHERS",
76                         "value": 2
77                     }
78                 ],
79                 "_owners": [
80                     {
81                         "key": 0,
82                         "value": 32
83                     },
84                     {
85                         "key": 16,
86                         "value": 128

```



```

87     },
88     {
89         "key": "ALL_OTHERS",
90         "value": 0
91     }
92 ],
93 "_operatorsForAll": [
94     {
95         "key": "ALL_OTHERS",
96         "value": [
97             {
98                 "key": "ALL_OTHERS",
99                 "value": false
100             }
101         ]
102     }
103 ],
104 "_operators": [
105     {
106         "key": 2,
107         "value": 1
108     },
109     {
110         "key": 16,
111         "value": 4
112     },
113     {
114         "key": "ALL_OTHERS",
115         "value": 0
116     }
117 ],
118 "_metaTransactionContracts": [
119     {
120         "key": "ALL_OTHERS",
121         "value": false
122     }
123 ],
124 "_admin": 0,
125 "_superOperators": [
126     {
127         "key": "ALL_OTHERS",
128         "value": true
129     }
130 ]
131 }
132 }
133 }
134 ]
135
136 Function invocation is reverted.

```

Formal Verification Request 268

__idInPath



10, Dec 2019



5.54 ms

Line 269-274 in File LandBaseToken.sol

```

269  /*@CTK _idInPath
270    @tag assume_completion
271    @pre GRID_SIZE == 408
272    @post (((i / size) % 2) == 0) -> __return == (x + (i%size)) + ((y + i / size) *
      GRID_SIZE)
273    @post (((i / size) % 2) == 1) -> __return == ((x + size) - (1 + i%size)) + ((y + i /
      size) * GRID_SIZE)
274  */

```

Line 275-282 in File LandBaseToken.sol

```

275  function _idInPath(uint256 i, uint256 size, uint256 x, uint256 y) internal pure returns(
    uint256) {
276    uint256 row = i / size;
277    if(row % 2 == 0) { // allow ids to follow a path in a quad
278      return (x + (i%size)) + ((y + row) * GRID_SIZE);
279    } else {
280      return ((x + size) - (1 + i%size)) + ((y + row) * GRID_SIZE);
281    }
282  }

```

✓ The code meets the specification.

Formal Verification Request 269

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 381.03 ms

Line 291 in File LandBaseToken.sol

```

291  //@CTK FAIL NO_OVERFLOW

```

Line 309-329 in File LandBaseToken.sol

```

309  function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
    bytes calldata data) external {
310    require(from != address(0), "from is zero address");
311    require(to != address(0), "can't send to zero address");
312    bool metaTx = msg.sender != from && !_metaTransactionContracts[msg.sender];
313    if (msg.sender != from && !metaTx) {
314      require(
315        _superOperators[msg.sender] ||
316        _operatorsForAll[from][msg.sender],
317        "not authorized to transferQuad"
318      );
319    }
320    _transferQuad(from, to, size, x, y);
321    _numNFTPerAddress[from] -= size * size;
322    _numNFTPerAddress[to] += size * size;
323
324    _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
    );
325  }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     data = ""
5     from = 64
6     size = 60
7     to = 1
8     x = 0
9     y = 0
10  }
11  This = 0
12  Internal = {
13    __has_assertion_failure = false
14    __has_buf_overflow = false
15    __has_overflow = false
16    __has_returned = false
17    __reverted = false
18    msg = {
19      "gas": 0,
20      "sender": 0,
21      "value": 0
22    }
23  }
24  Other = {
25    block = {
26      "number": 0,
27      "timestamp": 0
28    }
29  }
30  Address_Map = [
31    {
32      "key": 0,
33      "value": {
34        "contract_name": "LandBaseToken",
35        "balance": 0,
36        "contract": {
37          "GRID_SIZE": 0,
38          "LAYER": 0,
39          "LAYER_1x1": 0,
40          "LAYER_3x3": 0,
41          "LAYER_6x6": 0,
42          "LAYER_12x12": 0,
43          "LAYER_24x24": 0,
44          "_minters": [
45            {
46              "key": 128,
47              "value": true
48            },
49            {
50              "key": "ALL_OTHERS",
51              "value": false
52            }
53          ],
54          "_ERC721_RECEIVED": "}",
55          "_ERC721_BATCH_RECEIVED": "}",
56          "ERC165ID": "}",
57          "ERC721_MANDATORY_RECEIVER": "}"

```

```

58     "_numNFTPerAddress": [
59         {
60             "key": 65,
61             "value": 0
62         },
63         {
64             "key": 2,
65             "value": 0
66         },
67         {
68             "key": 0,
69             "value": 128
70         },
71         {
72             "key": 16,
73             "value": 2
74         },
75         {
76             "key": 128,
77             "value": 0
78         },
79         {
80             "key": 64,
81             "value": 0
82         },
83         {
84             "key": 1,
85             "value": 0
86         },
87         {
88             "key": "ALL_OTHERS",
89             "value": 60
90         }
91     ],
92     "_owners": [
93         {
94             "key": 2,
95             "value": 64
96         },
97         {
98             "key": 0,
99             "value": 0
100        },
101        {
102            "key": 1,
103            "value": 128
104        },
105        {
106            "key": "ALL_OTHERS",
107            "value": 60
108        }
109    ],
110    "_operatorsForAll": [
111        {
112            "key": "ALL_OTHERS",
113            "value": [
114                {
115                    "key": "ALL_OTHERS",

```

```

116         "value": false
117     }
118 ]
119 }
120 ],
121 "_operators": [
122     {
123         "key": 2,
124         "value": 64
125     },
126     {
127         "key": 0,
128         "value": 0
129     },
130     {
131         "key": 1,
132         "value": 128
133     },
134     {
135         "key": "ALL_OTHERS",
136         "value": 60
137     }
138 ],
139 "_metaTransactionContracts": [
140     {
141         "key": 0,
142         "value": true
143     },
144     {
145         "key": "ALL_OTHERS",
146         "value": false
147     }
148 ],
149 "_admin": 0,
150 "_superOperators": [
151     {
152         "key": 2,
153         "value": true
154     },
155     {
156         "key": 16,
157         "value": true
158     },
159     {
160         "key": "ALL_OTHERS",
161         "value": false
162     }
163 ]
164 }
165 },
166 {
167     "key": "ALL_OTHERS",
168     "value": "EmptyAddress"
169 }
170 ]
171 ]
172
173 After Execution:

```

```

174 Input = {
175     data = ""
176     from = 64
177     size = 60
178     to = 1
179     x = 0
180     y = 0
181 }
182 This = 0
183 Internal = {
184     __has_assertion_failure = false
185     __has_buf_overflow = false
186     __has_overflow = true
187     __has_returned = false
188     __reverted = false
189     msg = {
190         "gas": 0,
191         "sender": 0,
192         "value": 0
193     }
194 }
195 Other = {
196     block = {
197         "number": 0,
198         "timestamp": 0
199     }
200 }
201 Address_Map = [
202     {
203         "key": 0,
204         "value": {
205             "contract_name": "LandBaseToken",
206             "balance": 0,
207             "contract": {
208                 "GRID_SIZE": 0,
209                 "LAYER": 0,
210                 "LAYER_1x1": 0,
211                 "LAYER_3x3": 0,
212                 "LAYER_6x6": 0,
213                 "LAYER_12x12": 0,
214                 "LAYER_24x24": 0,
215                 "_minters": [
216                     {
217                         "key": 128,
218                         "value": true
219                     },
220                     {
221                         "key": "ALL_OTHERS",
222                         "value": false
223                     }
224                 ],
225                 "_ERC721_RECEIVED": "}}}}",
226                 "_ERC721_BATCH_RECEIVED": "}}}}",
227                 "ERC165ID": "}}}}",
228                 "ERC721_MANDATORY_RECEIVER": "}}}}",
229                 "_numNFTPerAddress": [
230                     {
231                         "key": 16,

```

```

232         "value": 2
233     },
234     {
235         "key": 2,
236         "value": 0
237     },
238     {
239         "key": 0,
240         "value": 128
241     },
242     {
243         "key": 65,
244         "value": 0
245     },
246     {
247         "key": 128,
248         "value": 0
249     },
250     {
251         "key": 64,
252         "value": 240
253     },
254     {
255         "key": 1,
256         "value": 16
257     },
258     {
259         "key": "ALL_OTHERS",
260         "value": 60
261     }
262 ],
263 "_owners": [
264     {
265         "key": 2,
266         "value": 64
267     },
268     {
269         "key": 0,
270         "value": 0
271     },
272     {
273         "key": 1,
274         "value": 128
275     },
276     {
277         "key": "ALL_OTHERS",
278         "value": 60
279     }
280 ],
281 "_operatorsForAll": [
282     {
283         "key": "ALL_OTHERS",
284         "value": [
285             {
286                 "key": "ALL_OTHERS",
287                 "value": false
288             }
289         ]

```

```

290     }
291 ],
292 "_operators": [
293   {
294     "key": 2,
295     "value": 64
296   },
297   {
298     "key": 0,
299     "value": 0
300   },
301   {
302     "key": 1,
303     "value": 128
304   },
305   {
306     "key": "ALL_OTHERS",
307     "value": 60
308   }
309 ],
310 "_metaTransactionContracts": [
311   {
312     "key": 0,
313     "value": true
314   },
315   {
316     "key": "ALL_OTHERS",
317     "value": false
318   }
319 ],
320 "_admin": 0,
321 "_superOperators": [
322   {
323     "key": 2,
324     "value": true
325   },
326   {
327     "key": 16,
328     "value": true
329   },
330   {
331     "key": "ALL_OTHERS",
332     "value": false
333   }
334 ]
335 }
336 }
337 },
338 {
339   "key": "ALL_OTHERS",
340   "value": "EmptyAddress"
341 }
342 ]

```


Formal Verification Request 270

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

📅 10, Dec 2019

🕒 16.05 ms

Line 292 in File LandBaseToken.sol

```
292  // @CTK_NO_BUF_OVERFLOW
```

Line 309-329 in File LandBaseToken.sol

```
309  function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
    bytes calldata data) external {
310      require(from != address(0), "from is zero address");
311      require(to != address(0), "can't send to zero address");
312      bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
313      if (msg.sender != from && !metaTx) {
314          require(
315              _superOperators[msg.sender] ||
316              _operatorsForAll[from][msg.sender],
317              "not authorized to transferQuad"
318          );
319      }
320      _transferQuad(from, to, size, x, y);
321      _numNFTPerAddress[from] -= size * size;
322      _numNFTPerAddress[to] += size * size;
323
324      _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
    );
325  }
```

✅ The code meets the specification.

Formal Verification Request 271

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 11.35 ms

Line 293 in File LandBaseToken.sol

```
293  // @CTK_NO_ASF
```

Line 309-329 in File LandBaseToken.sol

```
309  function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
    bytes calldata data) external {
310      require(from != address(0), "from is zero address");
311      require(to != address(0), "can't send to zero address");
312      bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
313      if (msg.sender != from && !metaTx) {
314          require(
315              _superOperators[msg.sender] ||
316              _operatorsForAll[from][msg.sender],
317              "not authorized to transferQuad"
318          );
    );
```

```

319     }
320     _transferQuad(from, to, size, x, y);
321     _numNFTPerAddress[from] -= size * size;
322     _numNFTPerAddress[to] += size * size;
323
324     _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
        );
325 }

```

✓ The code meets the specification.

Formal Verification Request 272

transferQuad_require

📅 10, Dec 2019

🕒 8.7 ms

Line 294-299 in File LandBaseToken.sol

```

294     /*@CTK transferQuad_require
295     @tag assume_completion
296     @post from != address(0)
297     @post to != address(0)
298     @post (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
        _superOperators[msg.sender] /\ _operatorsForAll[from][msg.sender])
299     */

```

Line 309-329 in File LandBaseToken.sol

```

309     function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
        bytes calldata data) external {
310         require(from != address(0), "from is zero address");
311         require(to != address(0), "can't send to zero address");
312         bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
313         if (msg.sender != from && !metaTx) {
314             require(
315                 _superOperators[msg.sender] ||
316                 _operatorsForAll[from][msg.sender],
317                 "not authorized to transferQuad"
318             );
319         }
320         _transferQuad(from, to, size, x, y);
321         _numNFTPerAddress[from] -= size * size;
322         _numNFTPerAddress[to] += size * size;
323
324         _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
            );
325     }


```

✓ The code meets the specification.

Formal Verification Request 273

transferQuad_change

📅 10, Dec 2019

 121.33 ms

Line 300-308 in File LandBaseToken.sol

```
300    /*@CTK transferQuad_change
301       @tag assume_completion
302       @pre from != to
303       @pre from != address(0)
304       @pre to != address(0)
305       @pre (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
           _superOperators[msg.sender] /\ _operatorsForAll[from][msg.sender])
306       @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - size * size
307       @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + size * size
308    */
```

Line 309-329 in File LandBaseToken.sol


```
309    function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
           bytes calldata data) external {
310        require(from != address(0), "from is zero address");
311        require(to != address(0), "can't send to zero address");
312        bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
313        if (msg.sender != from && !metaTx) {
314            require(
315                _superOperators[msg.sender] ||
316                _operatorsForAll[from][msg.sender],
317                "not authorized to transferQuad"
318            );
319        }
320        _transferQuad(from, to, size, x, y);
321        _numNFTPerAddress[from] -= size * size;
322        _numNFTPerAddress[to] += size * size;
323
324        _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
           );
325    }
```

 The code meets the specification.

Formal Verification Request 274

_checkBatchReceiverAcceptQuad

 10, Dec 2019

 92.23 ms

Line 331-335 in File LandBaseToken.sol

```
331    /*@CTK _checkBatchReceiverAcceptQuad
332       @tag assume_completion
333       @pre size >= 1
334       @pre GRID_SIZE == 408
335    */
```

Line 336-362 in File LandBaseToken.sol

```
336    function _checkBatchReceiverAcceptQuad(
337        address operator,
```

```

338     address from,
339     address to,
340     uint256 size,
341     uint256 x,
342     uint256 y,
343     bytes memory data
344 ) internal {
345     if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
346         uint256[] memory ids = new uint256[](size*size);
347         /*@CTK _checkBatchReceiverAcceptQuad_forloop
348             @inv i <= size * size
349             @pre size >= 1
350             @pre GRID_SIZE == 408
351             @post i == size * size
352             @post !__should_return
353             */
354         for (uint256 i = 0; i < size*size; i++) {
355             ids[i] = _idInPath(i, size, x, y);
356         }
357         require(
358             _checkOnERC721BatchReceived(operator, from, to, ids, data),
359             "erc721 batch transfer rejected by to"
360         );
361     }
362 }

```

✓ The code meets the specification.

Formal Verification Request 275

If method completes, integer overflow would not happen.



10, Dec 2019



62.47 ms

Line 371 in File LandBaseToken.sol

```
371 // @CTK NO_OVERFLOW
```

Line 380-427 in File LandBaseToken.sol

```

380     function batchTransferQuad(
381         address from,
382         address to,
383         uint256[] calldata sizes,
384         uint256[] calldata xs,
385         uint256[] calldata ys,
386         bytes calldata data
387     ) external {
388         require(from != address(0), "from is zero address");
389         require(to != address(0), "can't send to zero address");
390         require(sizes.length == xs.length && xs.length == ys.length, "invalid data");
391         bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
392         if (msg.sender != from && !metaTx) {
393             require(
394                 _superOperators[msg.sender] ||
395                 _operatorsForAll[from][msg.sender],
396                 "not authorized to transferMultiQuads"

```

```

397     );
398 }
399 uint256 numTokensTransferred = 0;
400 for (uint256 i = 0; i < sizes.length; i++) {
401     uint256 size = sizes[i];
402     _transferQuad(from, to, size, xs[i], ys[i]);
403     numTokensTransferred += size * size;
404 }
405 _numNFTPerAddress[from] -= numTokensTransferred;
406 _numNFTPerAddress[to] += numTokensTransferred;
407
408 if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
409     uint256[] memory ids = new uint256[](numTokensTransferred);
410     uint256 counter = 0;
411     for (uint256 j = 0; j < sizes.length; j++) {
412         uint256 size = sizes[j];
413         for (uint256 i = 0; i < size*size; i++) {
414             ids[counter] = _idInPath(i, size, xs[j], ys[j]);
415             counter++;
416         }
417     }
418     require(
419         _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
420         "erc721 batch transfer rejected by to"
421     );
422 }
423 }

```

✓ The code meets the specification.

Formal Verification Request 276

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



10, Dec 2019



19.59 ms

Line 372 in File LandBaseToken.sol

```
372 // @CTK NO_BUF_OVERFLOW
```

Line 380-427 in File LandBaseToken.sol

```

380 function batchTransferQuad(
381     address from,
382     address to,
383     uint256[] calldata sizes,
384     uint256[] calldata xs,
385     uint256[] calldata ys,
386     bytes calldata data
387 ) external {
388     require(from != address(0), "from is zero address");
389     require(to != address(0), "can't send to zero address");
390     require(sizes.length == xs.length && xs.length == ys.length, "invalid data");
391     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
392     if (msg.sender != from && !metaTx) {
393         require(
394             _superOperators[msg.sender] ||

```

```

395         _operatorsForAll[from][msg.sender],
396         "not authorized to transferMultiQuads"
397     );
398 }
399 uint256 numTokensTransferred = 0;
400 for (uint256 i = 0; i < sizes.length; i++) {
401     uint256 size = sizes[i];
402     _transferQuad(from, to, size, xs[i], ys[i]);
403     numTokensTransferred += size * size;
404 }
405 _numNFTPerAddress[from] -= numTokensTransferred;
406 _numNFTPerAddress[to] += numTokensTransferred;
407
408 if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
409     uint256[] memory ids = new uint256[](numTokensTransferred);
410     uint256 counter = 0;
411     for (uint256 j = 0; j < sizes.length; j++) {
412         uint256 size = sizes[j];
413         for (uint256 i = 0; i < size*size; i++) {
414             ids[counter] = _idInPath(i, size, xs[j], ys[j]);
415             counter++;
416         }
417     }
418     require(
419         _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
420         "erc721 batch transfer rejected by to"
421     );
422 }
423 }

```

✓ The code meets the specification.

Formal Verification Request 277

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 18.45 ms

Line 373 in File LandBaseToken.sol

```
373 // @CTK NO_ASF
```

Line 380-427 in File LandBaseToken.sol

```

380 function batchTransferQuad(
381     address from,
382     address to,
383     uint256[] calldata sizes,
384     uint256[] calldata xs,
385     uint256[] calldata ys,
386     bytes calldata data
387 ) external {
388     require(from != address(0), "from is zero address");
389     require(to != address(0), "can't send to zero address");
390     require(sizes.length == xs.length && xs.length == ys.length, "invalid data");
391     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
392     if (msg.sender != from && !metaTx) {

```

```

393         require(
394             _superOperators[msg.sender] ||
395             _operatorsForAll[from][msg.sender],
396             "not authorized to transferMultiQuads"
397         );
398     }
399     uint256 numTokensTransferred = 0;
400     for (uint256 i = 0; i < sizes.length; i++) {
401         uint256 size = sizes[i];
402         _transferQuad(from, to, size, xs[i], ys[i]);
403         numTokensTransferred += size * size;
404     }
405     _numNFTPerAddress[from] -= numTokensTransferred;
406     _numNFTPerAddress[to] += numTokensTransferred;
407
408     if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
409         uint256[] memory ids = new uint256[](numTokensTransferred);
410         uint256 counter = 0;
411         for (uint256 j = 0; j < sizes.length; j++) {
412             uint256 size = sizes[j];
413             for (uint256 i = 0; i < size*size; i++) {
414                 ids[counter] = _idInPath(i, size, xs[j], ys[j]);
415                 counter++;
416             }
417         }
418         require(
419             _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
420             "erc721 batch transfer rejected by to"
421         );
422     }
423 }

```

✓ The code meets the specification.

Formal Verification Request 278

transferQuad_require



10, Dec 2019



13.0 ms

Line 374-379 in File LandBaseToken.sol

```

374     /*@CTK transferQuad_require
375     @tag assume_completion
376     @post from != address(0)
377     @post to != address(0)
378     @post (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
379         _superOperators[msg.sender] /\ _operatorsForAll[from][msg.sender])
380     */

```

Line 380-427 in File LandBaseToken.sol

```

380     function batchTransferQuad(
381         address from,
382         address to,
383         uint256[] calldata sizes,
384         uint256[] calldata xs,

```

```

385     uint256[] calldata ys,
386     bytes calldata data
387 ) external {
388     require(from != address(0), "from is zero address");
389     require(to != address(0), "can't send to zero address");
390     require(sizes.length == xs.length && xs.length == ys.length, "invalid data");
391     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
392     if (msg.sender != from && !metaTx) {
393         require(
394             _superOperators[msg.sender] ||
395             _operatorsForAll[from][msg.sender],
396             "not authorized to transferMultiQuads"
397         );
398     }
399     uint256 numTokensTransferred = 0;
400     for (uint256 i = 0; i < sizes.length; i++) {
401         uint256 size = sizes[i];
402         _transferQuad(from, to, size, xs[i], ys[i]);
403         numTokensTransferred += size * size;
404     }
405     _numNFTPerAddress[from] -= numTokensTransferred;
406     _numNFTPerAddress[to] += numTokensTransferred;
407
408     if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
409         uint256[] memory ids = new uint256[](numTokensTransferred);
410         uint256 counter = 0;
411         for (uint256 j = 0; j < sizes.length; j++) {
412             uint256 size = sizes[j];
413             for (uint256 i = 0; i < size*size; i++) {
414                 ids[counter] = _idInPath(i, size, xs[j], ys[j]);
415                 counter++;
416             }
417         }
418         require(
419             _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
420             "erc721 batch transfer rejected by to"
421         );
422     }
423 }

```

✓ The code meets the specification.

Formal Verification Request 279

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



10, Dec 2019



100.86 ms

Line 429 in File LandBaseToken.sol

```
429 // @CTK NO_BUF_OVERFLOW
```

Line 442-461 in File LandBaseToken.sol

```

442 function _transferQuad(address from, address to, uint256 size, uint256 x, uint256 y)
443     internal {
444     if (size == 1) {

```



```

444     uint256 idx1 = x + y * GRID_SIZE;
445     address owner = _ownerOf(idx1);
446     require(owner != address(0), "token does not exist");
447     require(owner == from, "not owner in _transferQuad");
448     _owners[idx1] = uint256(to);
449 } else {
450     _regroup(from, to, size, x, y);
451 }
452 /*@CTK _transferQuad_loop
453   @inv i <= size * size
454   @post i == size * size
455   */
456 for (uint256 i = 0; i < size*size; i++) {
457     emit Transfer(from, to, _idInPath(i, size, x, y));
458 }
459 }

```

✓ The code meets the specification.

Formal Verification Request 280

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 1.63 ms

Line 430 in File LandBaseToken.sol

```
430 // @CTK NO_ASF
```

Line 442-461 in File LandBaseToken.sol

```

442 function _transferQuad(address from, address to, uint256 size, uint256 x, uint256 y)
443     internal {
444     if (size == 1) {
445         uint256 idx1 = x + y * GRID_SIZE;
446         address owner = _ownerOf(idx1);
447         require(owner != address(0), "token does not exist");
448         require(owner == from, "not owner in _transferQuad");
449         _owners[idx1] = uint256(to);
450     } else {
451         _regroup(from, to, size, x, y);
452     }
453     /*@CTK _transferQuad_loop
454       @inv i <= size * size
455       @post i == size * size
456       */
457     for (uint256 i = 0; i < size*size; i++) {
458         emit Transfer(from, to, _idInPath(i, size, x, y));
459     }
460 }

```

✓ The code meets the specification.

Formal Verification Request 281

If method completes, integer overflow would not happen.



10, Dec 2019



24.46 ms

Line 463 in File LandBaseToken.sol

```
463 // @CTK NO_OVERFLOW
```

Line 477-485 in File LandBaseToken.sol

```
477 function _checkAndClear(address from, uint256 id) internal returns(bool) {
478     uint256 owner = _owners[id];
479     if (owner != 0) {
480         require(address(owner) == from, "not owner");
481         _owners[id] = 0;
482         return true;
483     }
484     return false;
485 }
```



The code meets the specification.

Formal Verification Request 282

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



10, Dec 2019



0.52 ms

Line 464 in File LandBaseToken.sol

```
464 // @CTK NO_BUF_OVERFLOW
```

Line 477-485 in File LandBaseToken.sol

```
477 function _checkAndClear(address from, uint256 id) internal returns(bool) {
478     uint256 owner = _owners[id];
479     if (owner != 0) {
480         require(address(owner) == from, "not owner");
481         _owners[id] = 0;
482         return true;
483     }
484     return false;
485 }
```



The code meets the specification.

Formal Verification Request 283

Method will not encounter an assertion failure.



10, Dec 2019



0.5 ms

Line 465 in File LandBaseToken.sol

465 //CTK NO_ASF

Line 477-485 in File LandBaseToken.sol


```
477 function _checkAndClear(address from, uint256 id) internal returns(bool) {
478     uint256 owner = _owners[id];
479     if (owner != 0) {
480         require(address(owner) == from, "not owner");
481         _owners[id] = 0;
482         return true;
483     }
484     return false;
485 }
```

✓ The code meets the specification.

Formal Verification Request 284

`__checkAndClear__require`

 10, Dec 2019

 2.14 ms

Line 466-469 in File LandBaseToken.sol

```
466 /*CTK __checkAndClear__require
467     @tag assume_completion
468     @post (_owners[id] != 0) -> (address(_owners[id]) == from)
469 */
```

Line 477-485 in File LandBaseToken.sol


```
477 function _checkAndClear(address from, uint256 id) internal returns(bool) {
478     uint256 owner = _owners[id];
479     if (owner != 0) {
480         require(address(owner) == from, "not owner");
481         _owners[id] = 0;
482         return true;
483     }
484     return false;
485 }
```

✓ The code meets the specification.

Formal Verification Request 285

`__checkAndClear__change`

 10, Dec 2019

 2.69 ms

Line 470-476 in File LandBaseToken.sol

```
470 /*CTK __checkAndClear__change
471     @tag assume_completion
472     @pre (_owners[id] != 0) -> (address(_owners[id]) == from)
473     @post _owners[id] == 0 -> __return == false
```

```

474     @post (_owners[id] != 0) -> __post._owners[id] == 0
475     @post (_owners[id] != 0) -> __return == true
476     */

```

Line 477-485 in File LandBaseToken.sol

```

477     function _checkAndClear(address from, uint256 id) internal returns(bool) {
478         uint256 owner = _owners[id];
479         if (owner != 0) {
480             require(address(owner) == from, "not owner");
481             _owners[id] = 0;
482             return true;
483         }
484         return false;
485     }

```

✓ The code meets the specification.

Formal Verification Request 286

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 67.34 ms

Line 487 in File LandBaseToken.sol

```

487     //@CTK FAIL NO_OVERFLOW

```

Line 496-519 in File LandBaseToken.sol

```

496     function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
497     {
498         require(x % size == 0 && y % size == 0, "Invalid coordinates");
499         require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
500
501         if (size == 3) {
502             _regroup3x3(from, to, x, y, true);
503         } else if (size == 6) {
504             _regroup6x6(from, to, x, y, true);
505         } else if (size == 12) {
506             _regroup12x12(from, to, x, y, true);
507         } else if (size == 24) {
508             _regroup24x24(from, to, x, y, true);
509         } else {
510             require(false, "Invalid size");
511         }
512     }

```

✗ This code violates the specification.

1 Counter Example:

2 Before Execution:

```

3     Input = {
4         from = 0
5         size = 3
6         to = 0
7         x = 18
8         y = 153

```

```

9      }
10     This = 0
11     Internal = {
12         __has_assertion_failure = false
13         __has_buf_overflow = false
14         __has_overflow = false
15         __has_returned = false
16         __reverted = false
17         msg = {
18             "gas": 0,
19             "sender": 0,
20             "value": 0
21         }
22     }
23     Other = {
24         block = {
25             "number": 0,
26             "timestamp": 0
27         }
28     }
29     Address_Map = [
30         {
31             "key": "ALL_OTHERS",
32             "value": {
33                 "contract_name": "LandBaseToken",
34                 "balance": 0,
35                 "contract": {
36                     "GRID_SIZE": 0,
37                     "LAYER": 0,
38                     "LAYER_1x1": 0,
39                     "LAYER_3x3": 0,
40                     "LAYER_6x6": 0,
41                     "LAYER_12x12": 0,
42                     "LAYER_24x24": 0,
43                     "_minters": [
44                         {
45                             "key": "ALL_OTHERS",
46                             "value": false
47                         }
48                     ],
49                     "_ERC721_RECEIVED": "AAAA",
50                     "_ERC721_BATCH_RECEIVED": "AAAA",
51                     "ERC165ID": "AAAA",
52                     "ERC721_MANDATORY_RECEIVER": "CCCC",
53                     "_numNFTPerAddress": [
54                         {
55                             "key": 128,
56                             "value": 64
57                         },
58                         {
59                             "key": 0,
60                             "value": 32
61                         },
62                         {
63                             "key": 136,
64                             "value": 1
65                         },
66                     ]

```

```

67         "key": "ALL_OTHERS",
68         "value": 0
69     }
70 ],
71 "_owners": [
72     {
73         "key": 32,
74         "value": 64
75     },
76     {
77         "key": 96,
78         "value": 16
79     },
80     {
81         "key": 64,
82         "value": 8
83     },
84     {
85         "key": "ALL_OTHERS",
86         "value": 0
87     }
88 ],
89 "_operatorsForAll": [
90     {
91         "key": "ALL_OTHERS",
92         "value": [
93             {
94                 "key": "ALL_OTHERS",
95                 "value": false
96             }
97         ]
98     }
99 ],
100 "_operators": [
101     {
102         "key": 4,
103         "value": 16
104     },
105     {
106         "key": 64,
107         "value": 2
108     },
109     {
110         "key": 32,
111         "value": 128
112     },
113     {
114         "key": "ALL_OTHERS",
115         "value": 32
116     }
117 ],
118 "_metaTransactionContracts": [
119     {
120         "key": 0,
121         "value": false
122     },
123     {
124         "key": "ALL_OTHERS",

```

```

125         "value": true
126     }
127 ],
128     "_admin": 0,
129     "_superOperators": [
130     {
131         "key": "ALL_OTHERS",
132         "value": true
133     }
134 ]
135 }
136 }
137 }
138 ]
139
140 After Execution:
141 Input = {
142     from = 0
143     size = 3
144     to = 0
145     x = 18
146     y = 153
147 }
148 This = 0
149 Internal = {
150     __has_assertion_failure = false
151     __has_buf_overflow = false
152     __has_overflow = true
153     __has_returned = false
154     __reverted = false
155     msg = {
156         "gas": 0,
157         "sender": 0,
158         "value": 0
159     }
160 }
161 Other = {
162     block = {
163         "number": 0,
164         "timestamp": 0
165     }
166 }
167 Address_Map = [
168 {
169     "key": "ALL_OTHERS",
170     "value": {
171         "contract_name": "LandBaseToken",
172         "balance": 0,
173         "contract": {
174             "GRID_SIZE": 0,
175             "LAYER": 0,
176             "LAYER_1x1": 0,
177             "LAYER_3x3": 0,
178             "LAYER_6x6": 0,
179             "LAYER_12x12": 0,
180             "LAYER_24x24": 0,
181             "_minters": [
182

```

```

183         "key": "ALL_OTHERS",
184         "value": false
185     }
186 ],
187 "_ERC721_RECEIVED": "AAAA",
188 "_ERC721_BATCH_RECEIVED": "AAAA",
189 "ERC165ID": "AAAA",
190 "ERC721_MANDATORY_RECEIVER": "CCCC",
191 "_numNFTPerAddress": [
192     {
193         "key": 128,
194         "value": 64
195     },
196     {
197         "key": 0,
198         "value": 32
199     },
200     {
201         "key": 136,
202         "value": 1
203     },
204     {
205         "key": "ALL_OTHERS",
206         "value": 0
207     }
208 ],
209 "_owners": [
210     {
211         "key": 32,
212         "value": 64
213     },
214     {
215         "key": 96,
216         "value": 16
217     },
218     {
219         "key": 64,
220         "value": 8
221     },
222     {
223         "key": "ALL_OTHERS",
224         "value": 0
225     }
226 ],
227 "_operatorsForAll": [
228     {
229         "key": "ALL_OTHERS",
230         "value": [
231             {
232                 "key": "ALL_OTHERS",
233                 "value": false
234             }
235         ]
236     }
237 ],
238 "_operators": [
239     {
240         "key": 4,

```



```

241         "value": 16
242     },
243     {
244         "key": 64,
245         "value": 2
246     },
247     {
248         "key": 32,
249         "value": 128
250     },
251     {
252         "key": "ALL_OTHERS",
253         "value": 32
254     }
255 ],
256 "_metaTransactionContracts": [
257     {
258         "key": 0,
259         "value": false
260     },
261     {
262         "key": "ALL_OTHERS",
263         "value": true
264     }
265 ],
266 "_admin": 0,
267 "_superOperators": [
268     {
269         "key": "ALL_OTHERS",
270         "value": true
271     }
272 ]
273 }
274 }
275 }
276 ]

```

Formal Verification Request 287

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



10, Dec 2019



4.36 ms

Line 488 in File LandBaseToken.sol

```
488 // @CTK NO_BUF_OVERFLOW
```

Line 496-519 in File LandBaseToken.sol

```

496 function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
497 {
498     require(x % size == 0 && y % size == 0, "Invalid coordinates");
499     require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
500     if (size == 3) {
501         _regroup3x3(from, to, x, y, true);
502     } else if (size == 6) {

```

```

503     _regroup6x6(from, to, x, y, true);
504 } else if (size == 12) {
505     _regroup12x12(from, to, x, y, true);
506 } else if (size == 24) {
507     _regroup24x24(from, to, x, y, true);
508 } else {
509     require(false, "Invalid size");
510 }
511 }

```

✓ The code meets the specification.

Formal Verification Request 288

Method will not encounter an assertion failure.



10, Dec 2019



4.8 ms

Line 489 in File LandBaseToken.sol

```
489 //CTK NO_ASF
```

Line 496-519 in File LandBaseToken.sol

```

496 function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
497 {
498     require(x % size == 0 && y % size == 0, "Invalid coordinates");
499     require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
500
501     if (size == 3) {
502         _regroup3x3(from, to, x, y, true);
503     } else if (size == 6) {
504         _regroup6x6(from, to, x, y, true);
505     } else if (size == 12) {
506         _regroup12x12(from, to, x, y, true);
507     } else if (size == 24) {
508         _regroup24x24(from, to, x, y, true);
509     } else {
510         require(false, "Invalid size");
511     }
512 }

```

✓ The code meets the specification.

Formal Verification Request 289

__regroup__require



10, Dec 2019



8.1 ms

Line 490-495 in File LandBaseToken.sol

```

490 /*CTK _regroup_require
491 @tag assume_completion

```

```

492     @post (x % size == 0) /\ (y % size == 0)
493     @post (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
494     @post (size == 1 \/ size == 3 \/ size == 6 \/ size == 12 \/ size == 24)
495     */

```

Line 496-519 in File LandBaseToken.sol

```

496     function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
497     {
498         require(x % size == 0 && y % size == 0, "Invalid coordinates");
499         require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
500
501         if (size == 3) {
502             _regroup3x3(from, to, x, y, true);
503         } else if (size == 6) {
504             _regroup6x6(from, to, x, y, true);
505         } else if (size == 12) {
506             _regroup12x12(from, to, x, y, true);
507         } else if (size == 24) {
508             _regroup24x24(from, to, x, y, true);
509         } else {
510             require(false, "Invalid size");
511         }
512     }

```

✓ The code meets the specification.

Formal Verification Request 290

If method completes, integer overflow would not happen.



10, Dec 2019



2114.24 ms

Line 521 in File LandBaseToken.sol

```

521     //@CTK FAIL NO_OVERFLOW

```

Line 529-554 in File LandBaseToken.sol

```

529     function _regroup3x3(address from, address to, uint256 x, uint256 y, bool set) internal
530     returns (bool) {
531         uint256 id = x + y * GRID_SIZE;
532         uint256 quadId = LAYER_3x3 + id;
533         bool ownerOfAll = true;
534         for (uint256 xi = x; xi < x+3; xi++) {
535             for (uint256 yi = y; yi < y+3; yi++) {
536                 ownerOfAll = _checkAndClear(from, xi + yi * GRID_SIZE) && ownerOfAll;
537             }
538         }
539         if(set) {
540             if(!ownerOfAll) {
541                 require(
542                     _owners[quadId] == uint256(from) ||
543                     _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == uint256(from)
544                     ||
545                     _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
546                         from) ||

```

```

544         _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
           from),
545         "not owner of all sub quads nor parent quads"
546     );
547 }
548 _owners[quadId] = uint256(to);
549 return true;
550 }
551 return ownerOfAll;
552 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     from = 0
5     set = true
6     to = 0
7     x = 192
8     y = 27
9   }
10  This = 0
11  Internal = {
12    __has_assertion_failure = false
13    __has_buf_overflow = false
14    __has_overflow = false
15    __has_returned = false
16    __reverted = false
17    msg = {
18      "gas": 0,
19      "sender": 0,
20      "value": 0
21    }
22  }
23  Other = {
24    __return = false
25    block = {
26      "number": 0,
27      "timestamp": 0
28    }
29  }
30  Address_Map = [
31    {
32      "key": 0,
33      "value": {
34        "contract_name": "LandBaseToken",
35        "balance": 0,
36        "contract": {
37          "GRID_SIZE": 38,
38          "LAYER": 0,
39          "LAYER_1x1": 0,
40          "LAYER_3x3": 62,
41          "LAYER_6x6": 0,
42          "LAYER_12x12": 0,
43          "LAYER_24x24": 0,
44          "_minters": [
45            {
46              "key": "ALL_OTHERS",

```

```

47         "value": false
48     }
49 ],
50     "_ERC721_RECEIVED": "\u007f\u007f\u007f\u007f",
51     "_ERC721_BATCH_RECEIVED": "\u007f\u007f\u007f\u007f",
52     "ERC165ID": "\u007f\u007f\u007f\u007f",
53     "ERC721_MANDATORY_RECEIVER": "\u007f\u007f\u007f\u007f",
54     "_numNFTPerAddress": [
55     {
56         "key": 0,
57         "value": 0
58     },
59     {
60         "key": 64,
61         "value": 8
62     },
63     {
64         "key": 128,
65         "value": 32
66     },
67     {
68         "key": 4,
69         "value": 16
70     },
71     {
72         "key": 8,
73         "value": 0
74     },
75     {
76         "key": "ALL_OTHERS",
77         "value": 62
78     }
79 ],
80     "_owners": [
81     {
82         "key": 0,
83         "value": 1
84     },
85     {
86         "key": 128,
87         "value": 32
88     },
89     {
90         "key": 64,
91         "value": 8
92     },
93     {
94         "key": 8,
95         "value": 0
96     },
97     {
98         "key": 4,
99         "value": 16
100    },
101    {
102        "key": "ALL_OTHERS",
103        "value": 62
104    }

```

```

105 ],
106 "_operatorsForAll": [
107   {
108     "key": "ALL_OTHERS",
109     "value": [
110       {
111         "key": "ALL_OTHERS",
112         "value": false
113       }
114     ]
115   }
116 ],
117 "_operators": [
118   {
119     "key": 0,
120     "value": 64
121   },
122   {
123     "key": 64,
124     "value": 64
125   },
126   {
127     "key": 4,
128     "value": 32
129   },
130   {
131     "key": 8,
132     "value": 1
133   },
134   {
135     "key": "ALL_OTHERS",
136     "value": 62
137   }
138 ],
139 "_metaTransactionContracts": [
140   {
141     "key": 0,
142     "value": true
143   },
144   {
145     "key": 16,
146     "value": true
147   },
148   {
149     "key": "ALL_OTHERS",
150     "value": false
151   }
152 ],
153 "_admin": 0,
154 "_superOperators": [
155   {
156     "key": 32,
157     "value": true
158   },
159   {
160     "key": "ALL_OTHERS",
161     "value": false
162   }

```

```

163     ]
164   }
165 }
166 },
167 {
168   "key": "ALL_OTHERS",
169   "value": "EmptyAddress"
170 }
171 ]
172
173 After Execution:
174   Input = {
175     from = 0
176     set = true
177     to = 0
178     x = 192
179     y = 27
180   }
181   This = 0
182   Internal = {
183     __has_assertion_failure = false
184     __has_buf_overflow = false
185     __has_overflow = true
186     __has_returned = true
187     __reverted = false
188     msg = {
189       "gas": 0,
190       "sender": 0,
191       "value": 0
192     }
193   }
194   Other = {
195     __return = true
196     block = {
197       "number": 0,
198       "timestamp": 0
199     }
200   }
201   Address_Map = [
202     {
203       "key": 0,
204       "value": {
205         "contract_name": "LandBaseToken",
206         "balance": 0,
207         "contract": {
208           "GRID_SIZE": 38,
209           "LAYER": 0,
210           "LAYER_1x1": 0,
211           "LAYER_3x3": 62,
212           "LAYER_6x6": 0,
213           "LAYER_12x12": 0,
214           "LAYER_24x24": 0,
215           "_minters": [
216             {
217               "key": "ALL_OTHERS",
218               "value": false
219             }
220           ],

```

```

221     "_ERC721_RECEIVED": "\u007f\u007f\u007f\u007f",
222     "_ERC721_BATCH_RECEIVED": "\u007f\u007f\u007f\u007f",
223     "ERC165ID": "\u007f\u007f\u007f\u007f",
224     "ERC721_MANDATORY_RECEIVER": "\u007f\u007f\u007f\u007f",
225     "_numNFTPerAddress": [
226     {
227         "key": 0,
228         "value": 0
229     },
230     {
231         "key": 64,
232         "value": 8
233     },
234     {
235         "key": 128,
236         "value": 32
237     },
238     {
239         "key": 4,
240         "value": 16
241     },
242     {
243         "key": 8,
244         "value": 0
245     },
246     {
247         "key": "ALL_OTHERS",
248         "value": 62
249     }
250 ],
251     "_owners": [
252     {
253         "key": 0,
254         "value": 0
255     },
256     {
257         "key": 64,
258         "value": 8
259     },
260     {
261         "key": 128,
262         "value": 32
263     },
264     {
265         "key": 4,
266         "value": 16
267     },
268     {
269         "key": 8,
270         "value": 0
271     },
272     {
273         "key": "ALL_OTHERS",
274         "value": 62
275     }
276 ],
277     "_operatorsForAll": [
278     {

```



```

279         "key": "ALL_OTHERS",
280         "value": [
281             {
282                 "key": "ALL_OTHERS",
283                 "value": false
284             }
285         ]
286     },
287 ],
288 "_operators": [
289     {
290         "key": 0,
291         "value": 64
292     },
293     {
294         "key": 64,
295         "value": 64
296     },
297     {
298         "key": 4,
299         "value": 32
300     },
301     {
302         "key": 8,
303         "value": 1
304     },
305     {
306         "key": "ALL_OTHERS",
307         "value": 62
308     }
309 ],
310 "_metaTransactionContracts": [
311     {
312         "key": 0,
313         "value": true
314     },
315     {
316         "key": 16,
317         "value": true
318     },
319     {
320         "key": "ALL_OTHERS",
321         "value": false
322     }
323 ],
324 "_admin": 0,
325 "_superOperators": [
326     {
327         "key": 32,
328         "value": true
329     },
330     {
331         "key": "ALL_OTHERS",
332         "value": false
333     }
334 ]
335 }
336

```

```


337     },
338     {
339         "key": "ALL_OTHERS",
340         "value": "EmptyAddress"
341     }
342 ]

```

Formal Verification Request 291

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

 10, Dec 2019

 2.12 ms

Line 522 in File LandBaseToken.sol

```
522 // @CTK NO_BUF_OVERFLOW
```

Line 529-554 in File LandBaseToken.sol

```

529 function _regroup3x3(address from, address to, uint256 x, uint256 y, bool set) internal
    returns (bool) {
530     uint256 id = x + y * GRID_SIZE;
531     uint256 quadId = LAYER_3x3 + id;
532     bool ownerOfAll = true;
533     for (uint256 xi = x; xi < x+3; xi++) {
534         for (uint256 yi = y; yi < y+3; yi++) {
535             ownerOfAll = _checkAndClear(from, xi + yi * GRID_SIZE) && ownerOfAll;
536         }
537     }
538     if(set) {
539         if(!ownerOfAll) {
540             require(
541                 _owners[quadId] == uint256(from) ||
542                 _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == uint256(from)
543                 ||
544                 _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
545                     from) ||
546                 _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
547                     from),
548                 "not owner of all sub quads nor parent quads"
549             );
550         }
551         _owners[quadId] = uint256(to);
552         return true;
553     }
554     return ownerOfAll;
555 }


```

 The code meets the specification.

Formal Verification Request 292

Method will not encounter an assertion failure.

 10, Dec 2019

 1.43 ms

Line 523 in File LandBaseToken.sol

523 `//@CTK NO_ASF`

Line 529-554 in File LandBaseToken.sol

```

529  function _regroup3x3(address from, address to, uint256 x, uint256 y, bool set) internal
        returns (bool) {
530      uint256 id = x + y * GRID_SIZE;
531      uint256 quadId = LAYER_3x3 + id;
532      bool ownerOfAll = true;
533      for (uint256 xi = x; xi < x+3; xi++) {
534          for (uint256 yi = y; yi < y+3; yi++) {
535              ownerOfAll = _checkAndClear(from, xi + yi * GRID_SIZE) && ownerOfAll;
536          }
537      }
538      if(set) {
539          if(!ownerOfAll) {
540              require(
541                  _owners[quadId] == uint256(from) ||
542                  _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == uint256(from)
                    ||
543                  _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
                        from) ||
544                  _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
                        from),
                    "not owner of all sub quads nor parent quads"
                );
545          }
546          _owners[quadId] = uint256(to);
547          return true;
548      }
549      return ownerOfAll;
550  }
551  }
552  }
```

✓ The code meets the specification.

Formal Verification Request 293

If method completes, integer overflow would not happen.



10, Dec 2019



2044.49 ms

Line 556 in File LandBaseToken.sol

556 `//@CTK FAIL NO_OVERFLOW`

Line 559-592 in File LandBaseToken.sol

```

559  function _regroup6x6(address from, address to, uint256 x, uint256 y, bool set) internal
        returns (bool) {
560      uint256 id = x + y * GRID_SIZE;
561      uint256 quadId = LAYER_6x6 + id;
562      bool ownerOfAll = true;
563      for (uint256 xi = x; xi < x+6; xi += 3) {
564          for (uint256 yi = y; yi < y+6; yi += 3) {
565              bool ownAllIndividual = _regroup3x3(from, to, xi, yi, false);
```

```

566     uint256 id3x3 = LAYER_3x3 + xi + yi * GRID_SIZE;
567     uint256 owner3x3 = _owners[id3x3];
568     if (owner3x3 != 0) {
569         if(!ownAllIndividual) {
570             require(owner3x3 == uint256(from), "not owner of 3x3 quad");
571         }
572         _owners[id3x3] = 0;
573     }
574     ownerOfAll = (ownAllIndividual || owner3x3 != 0) && ownerOfAll;
575 }
576 }
577 if(set) {
578     if(!ownerOfAll) {
579         require(
580             _owners[quadId] == uint256(from) ||
581             _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
582                 from) ||
583             _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
584                 from),
585             "not owner of all sub quads nor parent quads"
586         );
587     }
588     _owners[quadId] = uint256(to);
589     return true;
590 }
591 return ownerOfAll;
592 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3     Input = {
4         from = 0
5         set = false
6         to = 0
7         x = 33
8         y = 163
9     }
10    This = 0
11    Internal = {
12        __has_assertion_failure = false
13        __has_buf_overflow = false
14        __has_overflow = false
15        __has_returned = false
16        __reverted = false
17        msg = {
18            "gas": 0,
19            "sender": 0,
20            "value": 0
21        }
22    }
23    Other = {
24        __return = false
25        block = {
26            "number": 0,
27            "timestamp": 0
28        }
29    }

```

```

30 Address_Map = [
31   {
32     "key": 0,
33     "value": {
34       "contract_name": "LandBaseToken",
35       "balance": 0,
36       "contract": {
37         "GRID_SIZE": 183,
38         "LAYER": 0,
39         "LAYER_1x1": 0,
40         "LAYER_3x3": 0,
41         "LAYER_6x6": 140,
42         "LAYER_12x12": 0,
43         "LAYER_24x24": 0,
44         "_minters": [
45           {
46             "key": 64,
47             "value": true
48           },
49           {
50             "key": 0,
51             "value": true
52           },
53           {
54             "key": 8,
55             "value": true
56           },
57           {
58             "key": "ALL_OTHERS",
59             "value": false
60           }
61         ],
62         "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
63         "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
64         "ERC165ID": "\u0081\u0081\u0081\u0081",
65         "ERC721_MANDATORY_RECEIVER": "\u0081\u0081\u0081\u0081",
66         "_numNFTPerAddress": [
67           {
68             "key": 68,
69             "value": 0
70           },
71           {
72             "key": 0,
73             "value": 8
74           },
75           {
76             "key": 4,
77             "value": 2
78           },
79           {
80             "key": "ALL_OTHERS",
81             "value": 64
82           }
83         ],
84         "_owners": [
85           {
86             "key": 0,
87             "value": 0

```

```

88     },
89     {
90         "key": 32,
91         "value": 128
92     },
93     {
94         "key": 64,
95         "value": 0
96     },
97     {
98         "key": "ALL_OTHERS",
99         "value": 64
100    }
101 ],
102 "_operatorsForAll": [
103     {
104         "key": 0,
105         "value": [
106             {
107                 "key": 0,
108                 "value": true
109             },
110             {
111                 "key": "ALL_OTHERS",
112                 "value": false
113             }
114         ]
115     },
116     {
117         "key": 1,
118         "value": [
119             {
120                 "key": 0,
121                 "value": true
122             },
123             {
124                 "key": "ALL_OTHERS",
125                 "value": false
126             }
127         ]
128     },
129     {
130         "key": "ALL_OTHERS",
131         "value": [
132             {
133                 "key": "ALL_OTHERS",
134                 "value": false
135             }
136         ]
137     }
138 ],
139 "_operators": [
140     {
141         "key": 0,
142         "value": 0
143     },
144     {
145         "key": 32,

```

```

146         "value": 128
147     },
148     {
149         "key": 64,
150         "value": 0
151     },
152     {
153         "key": "ALL_OTHERS",
154         "value": 64
155     }
156 ],
157 "_metaTransactionContracts": [
158     {
159         "key": 2,
160         "value": true
161     },
162     {
163         "key": 4,
164         "value": true
165     },
166     {
167         "key": "ALL_OTHERS",
168         "value": false
169     }
170 ],
171 "_admin": 0,
172 "_superOperators": [
173     {
174         "key": 1,
175         "value": true
176     },
177     {
178         "key": "ALL_OTHERS",
179         "value": false
180     }
181 ]
182 }
183 }
184 },
185 {
186     "key": "ALL_OTHERS",
187     "value": "EmptyAddress"
188 }
189 ]

```

191 After Execution:

```

192     Input = {
193         from = 0
194         set = false
195         to = 0
196         x = 33
197         y = 163
198     }
199     This = 0
200     Internal = {
201         __has_assertion_failure = false
202         __has_buf_overflow = false
203         __has_overflow = true

```

```

204     __has_returned = true
205     __reverted = false
206     msg = {
207         "gas": 0,
208         "sender": 0,
209         "value": 0
210     }
211 }
212 Other = {
213     __return = true
214     block = {
215         "number": 0,
216         "timestamp": 0
217     }
218 }
219 Address_Map = [
220     {
221         "key": 0,
222         "value": {
223             "contract_name": "LandBaseToken",
224             "balance": 0,
225             "contract": {
226                 "GRID_SIZE": 183,
227                 "LAYER": 0,
228                 "LAYER_1x1": 0,
229                 "LAYER_3x3": 0,
230                 "LAYER_6x6": 140,
231                 "LAYER_12x12": 0,
232                 "LAYER_24x24": 0,
233                 "_minters": [
234                     {
235                         "key": 64,
236                         "value": true
237                     },
238                     {
239                         "key": 0,
240                         "value": true
241                     },
242                     {
243                         "key": 8,
244                         "value": true
245                     },
246                     {
247                         "key": "ALL_OTHERS",
248                         "value": false
249                     }
250                 ],
251                 "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
252                 "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
253                 "ERC165ID": "\u0081\u0081\u0081\u0081",
254                 "ERC721_MANDATORY_RECEIVER": "\u0081\u0081\u0081\u0081",
255                 "_numNFTPerAddress": [
256                     {
257                         "key": 68,
258                         "value": 0
259                     },
260                     {
261                         "key": 0,

```



```

262         "value": 8
263     },
264     {
265         "key": 4,
266         "value": 2
267     },
268     {
269         "key": "ALL_OTHERS",
270         "value": 64
271     }
272 ],
273 "_owners": [
274     {
275         "key": 0,
276         "value": 0
277     },
278     {
279         "key": 32,
280         "value": 128
281     },
282     {
283         "key": 64,
284         "value": 0
285     },
286     {
287         "key": "ALL_OTHERS",
288         "value": 64
289     }
290 ],
291 "_operatorsForAll": [
292     {
293         "key": 0,
294         "value": [
295             {
296                 "key": 0,
297                 "value": true
298             },
299             {
300                 "key": "ALL_OTHERS",
301                 "value": false
302             }
303         ]
304     },
305     {
306         "key": 1,
307         "value": [
308             {
309                 "key": 0,
310                 "value": true
311             },
312             {
313                 "key": "ALL_OTHERS",
314                 "value": false
315             }
316         ]
317     },
318     {
319         "key": "ALL_OTHERS",

```

```

320         "value": [
321             {
322                 "key": "ALL_OTHERS",
323                 "value": false
324             }
325         ]
326     },
327 ],
328 "_operators": [
329     {
330         "key": 0,
331         "value": 0
332     },
333     {
334         "key": 32,
335         "value": 128
336     },
337     {
338         "key": 64,
339         "value": 0
340     },
341     {
342         "key": "ALL_OTHERS",
343         "value": 64
344     }
345 ],
346 "_metaTransactionContracts": [
347     {
348         "key": 2,
349         "value": true
350     },
351     {
352         "key": 4,
353         "value": true
354     },
355     {
356         "key": "ALL_OTHERS",
357         "value": false
358     }
359 ],
360 "_admin": 0,
361 "_superOperators": [
362     {
363         "key": 1,
364         "value": true
365     },
366     {
367         "key": "ALL_OTHERS",
368         "value": false
369     }
370 ],
371 ]
372 }
373 },
374 {
375     "key": "ALL_OTHERS",
376     "value": "EmptyAddress"
377 }

```

378]

Formal Verification Request 294

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



10, Dec 2019



2.12 ms

Line 557 in File LandBaseToken.sol

557 // @CTK NO_BUF_OVERFLOW

Line 559-592 in File LandBaseToken.sol

```

559 function _regroup6x6(address from, address to, uint256 x, uint256 y, bool set) internal
    returns (bool) {
560     uint256 id = x + y * GRID_SIZE;
561     uint256 quadId = LAYER_6x6 + id;
562     bool ownerOfAll = true;
563     for (uint256 xi = x; xi < x+6; xi += 3) {
564         for (uint256 yi = y; yi < y+6; yi += 3) {
565             bool ownAllIndividual = _regroup3x3(from, to, xi, yi, false);
566             uint256 id3x3 = LAYER_3x3 + xi + yi * GRID_SIZE;
567             uint256 owner3x3 = _owners[id3x3];
568             if (owner3x3 != 0) {
569                 if (!ownAllIndividual) {
570                     require(owner3x3 == uint256(from), "not owner of 3x3 quad");
571                 }
572                 _owners[id3x3] = 0;
573             }
574             ownerOfAll = (ownAllIndividual || owner3x3 != 0) && ownerOfAll;
575         }
576     }
577     if (set) {
578         if (!ownerOfAll) {
579             require(
580                 _owners[quadId] == uint256(from) ||
581                 _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
582                     from) ||
583                 _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
584                     from),
585                 "not owner of all sub quads nor parent quads"
586             );
587             _owners[quadId] = uint256(to);
588             return true;
589         }
590     }
    return ownerOfAll;
}

```

✓ The code meets the specification.

Formal Verification Request 295

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 1.24 ms

Line 558 in File LandBaseToken.sol

558 `//@CTK NO_ASF`

Line 559-592 in File LandBaseToken.sol

```

559     function _regroup6x6(address from, address to, uint256 x, uint256 y, bool set) internal
        returns (bool) {
560         uint256 id = x + y * GRID_SIZE;
561         uint256 quadId = LAYER_6x6 + id;
562         bool ownerOfAll = true;
563         for (uint256 xi = x; xi < x+6; xi += 3) {
564             for (uint256 yi = y; yi < y+6; yi += 3) {
565                 bool ownAllIndividual = _regroup3x3(from, to, xi, yi, false);
566                 uint256 id3x3 = LAYER_3x3 + xi + yi * GRID_SIZE;
567                 uint256 owner3x3 = _owners[id3x3];
568                 if (owner3x3 != 0) {
569                     if(!ownAllIndividual) {
570                         require(owner3x3 == uint256(from), "not owner of 3x3 quad");
571                     }
572                     _owners[id3x3] = 0;
573                 }
574                 ownerOfAll = (ownAllIndividual || owner3x3 != 0) && ownerOfAll;
575             }
576         }
577         if(set) {
578             if(!ownerOfAll) {
579                 require(
580                     _owners[quadId] == uint256(from) ||
581                     _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
582                         from) ||
583                     _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
584                         from),
585                     "not owner of all sub quads nor parent quads"
586                 );
587             }
588             _owners[quadId] = uint256(to);
589             return true;
590         }
591         return ownerOfAll;
592     }

```

✅ The code meets the specification.

Formal Verification Request 296

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 1035.14 ms

Line 594 in File LandBaseToken.sol

594 `//@CTK FAIL NO_OVERFLOW`

Line 597-629 in File LandBaseToken.sol

```
597 function _regroup12x12(address from, address to, uint256 x, uint256 y, bool set)
    internal returns (bool) {
598     uint256 id = x + y * GRID_SIZE;
599     uint256 quadId = LAYER_12x12 + id;
600     bool ownerOfAll = true;
601     for (uint256 xi = x; xi < x+12; xi += 6) {
602         for (uint256 yi = y; yi < y+12; yi += 6) {
603             bool ownAllIndividual = _regroup6x6(from, to, xi, yi, false);
604             uint256 id6x6 = LAYER_6x6 + xi + yi * GRID_SIZE;
605             uint256 owner6x6 = _owners[id6x6];
606             if (owner6x6 != 0) {
607                 if(!ownAllIndividual) {
608                     require(owner6x6 == uint256(from), "not owner of 6x6 quad");
609                 }
610                 _owners[id6x6] = 0;
611             }
612             ownerOfAll = (ownAllIndividual || owner6x6 != 0) && ownerOfAll;
613         }
614     }
615     if(set) {
616         if(!ownerOfAll) {
617             require(
618                 _owners[quadId] == uint256(from) ||
619                 _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
620                     from),
621                 "not owner of all sub quads nor parent quads"
622             );
623             _owners[quadId] = uint256(to);
624             return true;
625         }
626         return ownerOfAll;
627     }
```

✗ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3     Input = {
4         from = 0
5         set = false
6         to = 0
7         x = 8
8         y = 113
9     }
10 This = 0
11 Internal = {
12     __has_assertion_failure = false
13     __has_buf_overflow = false
14     __has_overflow = false
15     __has_returned = false
16     __reverted = false
17     msg = {
```

```

18     "gas": 0,
19     "sender": 0,
20     "value": 0
21 }
22 }
23 Other = {
24     __return = false
25     block = {
26         "number": 0,
27         "timestamp": 0
28     }
29 }
30 Address_Map = [
31     {
32         "key": 0,
33         "value": {
34             "contract_name": "LandBaseToken",
35             "balance": 0,
36             "contract": {
37                 "GRID_SIZE": 17,
38                 "LAYER": 0,
39                 "LAYER_1x1": 0,
40                 "LAYER_3x3": 0,
41                 "LAYER_6x6": 0,
42                 "LAYER_12x12": 130,
43                 "LAYER_24x24": 0,
44                 "_minters": [
45                     {
46                         "key": "ALL_OTHERS",
47                         "value": false
48                     }
49                 ],
50                 "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
51                 "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
52                 "ERC165ID": "\u0081\u0081\u0081\u0081",
53                 "ERC721_MANDATORY_RECEIVER": "\u00c2\u00c2\u00c2\u00c2",
54                 "_numNFTPerAddress": [
55                     {
56                         "key": 18,
57                         "value": 32
58                     },
59                     {
60                         "key": 0,
61                         "value": 0
62                     },
63                     {
64                         "key": 4,
65                         "value": 2
66                     },
67                     {
68                         "key": 8,
69                         "value": 8
70                     },
71                     {
72                         "key": "ALL_OTHERS",
73                         "value": 64
74                     }
75                 ],

```

```

76     "_owners": [
77         {
78             "key": 2,
79             "value": 72
80         },
81         {
82             "key": 0,
83             "value": 8
84         },
85         {
86             "key": 32,
87             "value": 0
88         },
89         {
90             "key": 128,
91             "value": 0
92         },
93         {
94             "key": 16,
95             "value": 0
96         },
97         {
98             "key": "ALL_OTHERS",
99             "value": 64
100     }
101 ],
102 "_operatorsForAll": [
103     {
104         "key": 0,
105         "value": [
106             {
107                 "key": 0,
108                 "value": true
109             },
110             {
111                 "key": "ALL_OTHERS",
112                 "value": false
113             }
114         ]
115     },
116     {
117         "key": 1,
118         "value": [
119             {
120                 "key": 0,
121                 "value": true
122             },
123             {
124                 "key": "ALL_OTHERS",
125                 "value": false
126             }
127         ]
128     },
129     {
130         "key": "ALL_OTHERS",
131         "value": [
132             {
133                 "key": "ALL_OTHERS",

```

```

134         "value": false
135     }
136 ]
137 }
138 ],
139 "_operators": [
140     {
141         "key": 0,
142         "value": 0
143     },
144     {
145         "key": 64,
146         "value": 32
147     },
148     {
149         "key": 8,
150         "value": 0
151     },
152     {
153         "key": 1,
154         "value": 4
155     },
156     {
157         "key": "ALL_OTHERS",
158         "value": 64
159     }
160 ],
161 "_metaTransactionContracts": [
162     {
163         "key": "ALL_OTHERS",
164         "value": false
165     }
166 ],
167 "_admin": 0,
168 "_superOperators": [
169     {
170         "key": "ALL_OTHERS",
171         "value": false
172     }
173 ]
174 }
175 }
176 },
177 {
178     "key": "ALL_OTHERS",
179     "value": "EmptyAddress"
180 }
181 ]

```

183 After Execution:

```

184     Input = {
185         from = 0
186         set = false
187         to = 0
188         x = 8
189         y = 113
190     }
191     This = 0

```



```

192 Internal = {
193     __has_assertion_failure = false
194     __has_buf_overflow = false
195     __has_overflow = true
196     __has_returned = true
197     __reverted = false
198     msg = {
199         "gas": 0,
200         "sender": 0,
201         "value": 0
202     }
203 }
204 Other = {
205     __return = true
206     block = {
207         "number": 0,
208         "timestamp": 0
209     }
210 }
211 Address_Map = [
212     {
213         "key": 0,
214         "value": {
215             "contract_name": "LandBaseToken",
216             "balance": 0,
217             "contract": {
218                 "GRID_SIZE": 17,
219                 "LAYER": 0,
220                 "LAYER_1x1": 0,
221                 "LAYER_3x3": 0,
222                 "LAYER_6x6": 0,
223                 "LAYER_12x12": 130,
224                 "LAYER_24x24": 0,
225                 "_minters": [
226                     {
227                         "key": "ALL_OTHERS",
228                         "value": false
229                     }
230                 ],
231                 "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
232                 "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
233                 "ERC165ID": "\u0081\u0081\u0081\u0081",
234                 "ERC721_MANDATORY_RECEIVER": "\u00c2\u00c2\u00c2\u00c2",
235                 "_numNFTPerAddress": [
236                     {
237                         "key": 18,
238                         "value": 32
239                     },
240                     {
241                         "key": 0,
242                         "value": 0
243                     },
244                     {
245                         "key": 4,
246                         "value": 2
247                     },
248                     {
249                         "key": 8,

```

```

250         "value": 8
251     },
252     {
253         "key": "ALL_OTHERS",
254         "value": 64
255     }
256 ],
257 "_owners": [
258     {
259         "key": 2,
260         "value": 72
261     },
262     {
263         "key": 0,
264         "value": 8
265     },
266     {
267         "key": 32,
268         "value": 0
269     },
270     {
271         "key": 128,
272         "value": 0
273     },
274     {
275         "key": 16,
276         "value": 0
277     },
278     {
279         "key": "ALL_OTHERS",
280         "value": 64
281     }
282 ],
283 "_operatorsForAll": [
284     {
285         "key": 0,
286         "value": [
287             {
288                 "key": 0,
289                 "value": true
290             },
291             {
292                 "key": "ALL_OTHERS",
293                 "value": false
294             }
295         ]
296     },
297     {
298         "key": 1,
299         "value": [
300             {
301                 "key": 0,
302                 "value": true
303             },
304             {
305                 "key": "ALL_OTHERS",
306                 "value": false
307             }

```

```

308     ]
309   },
310   {
311     "key": "ALL_OTHERS",
312     "value": [
313       {
314         "key": "ALL_OTHERS",
315         "value": false
316       }
317     ]
318   }
319 ],
320 "_operators": [
321   {
322     "key": 0,
323     "value": 0
324   },
325   {
326     "key": 64,
327     "value": 32
328   },
329   {
330     "key": 8,
331     "value": 0
332   },
333   {
334     "key": 1,
335     "value": 4
336   },
337   {
338     "key": "ALL_OTHERS",
339     "value": 64
340   }
341 ],
342 "_metaTransactionContracts": [
343   {
344     "key": "ALL_OTHERS",
345     "value": false
346   }
347 ],
348 "_admin": 0,
349 "_superOperators": [
350   {
351     "key": "ALL_OTHERS",
352     "value": false
353   }
354 ]
355 }
356 }
357 },
358 {
359   "key": "ALL_OTHERS",
360   "value": "EmptyAddress"
361 }
362 ]

```

Formal Verification Request 297

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

📅 10, Dec 2019

🕒 2.35 ms

Line 595 in File LandBaseToken.sol

595 `//@CTK NO_BUF_OVERFLOW`

Line 597-629 in File LandBaseToken.sol

```

597  function _regroup12x12(address from, address to, uint256 x, uint256 y, bool set)
598      internal returns (bool) {
599      uint256 id = x + y * GRID_SIZE;
600      uint256 quadId = LAYER_12x12 + id;
601      bool ownerOfAll = true;
602      for (uint256 xi = x; xi < x+12; xi += 6) {
603          for (uint256 yi = y; yi < y+12; yi += 6) {
604              bool ownAllIndividual = _regroup6x6(from, to, xi, yi, false);
605              uint256 id6x6 = LAYER_6x6 + xi + yi * GRID_SIZE;
606              uint256 owner6x6 = _owners[id6x6];
607              if (owner6x6 != 0) {
608                  if(!ownAllIndividual) {
609                      require(owner6x6 == uint256(from), "not owner of 6x6 quad");
610                  }
611                  _owners[id6x6] = 0;
612              }
613              ownAllIndividual = (ownAllIndividual || owner6x6 != 0) && ownerOfAll;
614          }
615      }
616      if(set) {
617          if(!ownerOfAll) {
618              require(
619                  _owners[quadId] == uint256(from) ||
620                  _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
621                      from),
622                  "not owner of all sub quads nor parent quads"
623              );
624              _owners[quadId] = uint256(to);
625              return true;
626          }
627      }
        
```

✅ The code meets the specification.

Formal Verification Request 298

Method will not encounter an assertion failure.

📅 10, Dec 2019

🕒 1.07 ms

Line 596 in File LandBaseToken.sol

596 //CTK NO_ASF

Line 597-629 in File LandBaseToken.sol

```

597 function _regroup12x12(address from, address to, uint256 x, uint256 y, bool set)
    internal returns (bool) {
598     uint256 id = x + y * GRID_SIZE;
599     uint256 quadId = LAYER_12x12 + id;
600     bool ownerOfAll = true;
601     for (uint256 xi = x; xi < x+12; xi += 6) {
602         for (uint256 yi = y; yi < y+12; yi += 6) {
603             bool ownAllIndividual = _regroup6x6(from, to, xi, yi, false);
604             uint256 id6x6 = LAYER_6x6 + xi + yi * GRID_SIZE;
605             uint256 owner6x6 = _owners[id6x6];
606             if (owner6x6 != 0) {
607                 if(!ownAllIndividual) {
608                     require(owner6x6 == uint256(from), "not owner of 6x6 quad");
609                 }
610                 _owners[id6x6] = 0;
611             }
612             ownerOfAll = (ownAllIndividual || owner6x6 != 0) && ownerOfAll;
613         }
614     }
615     if(set) {
616         if(!ownerOfAll) {
617             require(
618                 _owners[quadId] == uint256(from) ||
619                 _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
620                     from),
621                 "not owner of all sub quads nor parent quads"
622             );
623             _owners[quadId] = uint256(to);
624             return true;
625         }
626         return ownerOfAll;
627     }

```

✓ The code meets the specification.

Formal Verification Request 299

If method completes, integer overflow would not happen.



10, Dec 2019



3633.67 ms

Line 631 in File LandBaseToken.sol

631 //CTK FAIL NO_OVERFLOW

Line 634-665 in File LandBaseToken.sol

```

634 function _regroup24x24(address from, address to, uint256 x, uint256 y, bool set)
    internal returns (bool) {
635     uint256 id = x + y * GRID_SIZE;
636     uint256 quadId = LAYER_24x24 + id;
637     bool ownerOfAll = true;

```

```

638     for (uint256 xi = x; xi < x+24; xi += 12) {
639         for (uint256 yi = y; yi < y+24; yi += 12) {
640             bool ownAllIndividual = _regroup12x12(from, to, xi, yi, false);
641             uint256 id12x12 = LAYER_12x12 + xi + yi * GRID_SIZE;
642             uint256 owner12x12 = _owners[id12x12];
643             if (owner12x12 != 0) {
644                 if(!ownAllIndividual) {
645                     require(owner12x12 == uint256(from), "not owner of 12x12 quad");
646                 }
647                 _owners[id12x12] = 0;
648             }
649             ownerOfAll = (ownAllIndividual || owner12x12 != 0) && ownerOfAll;
650         }
651     }
652     if(set) {
653         if(!ownerOfAll) {
654             require(
655                 _owners[quadId] == uint256(from),
656                 "not owner of all sub quads not parent quad"
657             );
658         }
659         _owners[quadId] = uint256(to);
660         return true;
661     }
662     return ownerOfAll || _owners[quadId] == uint256(from);
663 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3     Input = {
4         from = 0
5         set = false
6         to = 0
7         x = 128
8         y = 0
9     }
10    This = 0
11    Internal = {
12        __has_assertion_failure = false
13        __has_buf_overflow = false
14        __has_overflow = false
15        __has_returned = false
16        __reverted = false
17        msg = {
18            "gas": 0,
19            "sender": 0,
20            "value": 0
21        }
22    }
23    Other = {
24        __return = false
25        block = {
26            "number": 0,
27            "timestamp": 0
28        }
29    }
30    Address_Map = [

```

```

31 {
32   "key": 0,
33   "value": {
34     "contract_name": "LandBaseToken",
35     "balance": 0,
36     "contract": {
37       "GRID_SIZE": 0,
38       "LAYER": 0,
39       "LAYER_1x1": 0,
40       "LAYER_3x3": 0,
41       "LAYER_6x6": 0,
42       "LAYER_12x12": 0,
43       "LAYER_24x24": 128,
44       "_minters": [
45         {
46           "key": "ALL_OTHERS",
47           "value": false
48         }
49       ],
50       "_ERC721_RECEIVED": "AAAA",
51       "_ERC721_BATCH_RECEIVED": "CCCC",
52       "ERC165ID": "CCCC",
53       "ERC721_MANDATORY_RECEIVER": "CCCC",
54       "_numNFTPerAddress": [
55         {
56           "key": 64,
57           "value": 64
58         },
59         {
60           "key": 8,
61           "value": 32
62         },
63         {
64           "key": 1,
65           "value": 4
66         },
67         {
68           "key": 9,
69           "value": 16
70         },
71         {
72           "key": 0,
73           "value": 1
74         },
75         {
76           "key": "ALL_OTHERS",
77           "value": 2
78         }
79       ],
80       "_owners": [
81         {
82           "key": 4,
83           "value": 0
84         },
85         {
86           "key": 32,
87           "value": 0
88         },

```

```

89     {
90         "key": 16,
91         "value": 16
92     },
93     {
94         "key": 0,
95         "value": 16
96     },
97     {
98         "key": "ALL_OTHERS",
99         "value": 2
100    }
101 ],
102 "_operatorsForAll": [
103     {
104         "key": 8,
105         "value": [
106             {
107                 "key": 0,
108                 "value": true
109             },
110             {
111                 "key": "ALL_OTHERS",
112                 "value": false
113             }
114         ]
115     },
116     {
117         "key": 0,
118         "value": [
119             {
120                 "key": 0,
121                 "value": true
122             },
123             {
124                 "key": "ALL_OTHERS",
125                 "value": false
126             }
127         ]
128     },
129     {
130         "key": "ALL_OTHERS",
131         "value": [
132             {
133                 "key": "ALL_OTHERS",
134                 "value": false
135             }
136         ]
137     }
138 ],
139 "_operators": [
140     {
141         "key": 4,
142         "value": 8
143     },
144     {
145         "key": 32,
146         "value": 0

```



```

147     },
148     {
149         "key": 64,
150         "value": 0
151     },
152     {
153         "key": 2,
154         "value": 8
155     },
156     {
157         "key": 0,
158         "value": 4
159     },
160     {
161         "key": "ALL_OTHERS",
162         "value": 2
163     }
164 ],
165 "_metaTransactionContracts": [
166     {
167         "key": 32,
168         "value": true
169     },
170     {
171         "key": "ALL_OTHERS",
172         "value": false
173     }
174 ],
175 "_admin": 0,
176 "_superOperators": [
177     {
178         "key": "ALL_OTHERS",
179         "value": false
180     }
181 ]
182 }
183 }
184 },
185 {
186     "key": "ALL_OTHERS",
187     "value": "EmptyAddress"
188 }
189 ]

```

191 After Execution:

```

192     Input = {
193         from = 0
194         set = false
195         to = 0
196         x = 128
197         y = 0
198     }
199     This = 0
200     Internal = {
201         __has_assertion_failure = false
202         __has_buf_overflow = false
203         __has_overflow = true
204         __has_returned = true

```

```

205     __reverted = false
206     msg = {
207         "gas": 0,
208         "sender": 0,
209         "value": 0
210     }
211 }
212 Other = {
213     __return = true
214     block = {
215         "number": 0,
216         "timestamp": 0
217     }
218 }
219 Address_Map = [
220     {
221         "key": 0,
222         "value": {
223             "contract_name": "LandBaseToken",
224             "balance": 0,
225             "contract": {
226                 "GRID_SIZE": 0,
227                 "LAYER": 0,
228                 "LAYER_1x1": 0,
229                 "LAYER_3x3": 0,
230                 "LAYER_6x6": 0,
231                 "LAYER_12x12": 0,
232                 "LAYER_24x24": 128,
233                 "_minters": [
234                     {
235                         "key": "ALL_OTHERS",
236                         "value": false
237                     }
238                 ],
239                 "_ERC721_RECEIVED": "AAAA",
240                 "_ERC721_BATCH_RECEIVED": "CCCC",
241                 "ERC165ID": "CCCC",
242                 "ERC721_MANDATORY_RECEIVER": "CCCC",
243                 "_numNFTPerAddress": [
244                     {
245                         "key": 64,
246                         "value": 64
247                     },
248                     {
249                         "key": 8,
250                         "value": 32
251                     },
252                     {
253                         "key": 1,
254                         "value": 4
255                     },
256                     {
257                         "key": 9,
258                         "value": 16
259                     },
260                     {
261                         "key": 0,
262                         "value": 1

```

```

263     },
264     {
265         "key": "ALL_OTHERS",
266         "value": 2
267     }
268 ],
269 "_owners": [
270     {
271         "key": 4,
272         "value": 0
273     },
274     {
275         "key": 32,
276         "value": 0
277     },
278     {
279         "key": 16,
280         "value": 16
281     },
282     {
283         "key": 0,
284         "value": 16
285     },
286     {
287         "key": "ALL_OTHERS",
288         "value": 2
289     }
290 ],
291 "_operatorsForAll": [
292     {
293         "key": 8,
294         "value": [
295             {
296                 "key": 0,
297                 "value": true
298             },
299             {
300                 "key": "ALL_OTHERS",
301                 "value": false
302             }
303         ]
304     },
305     {
306         "key": 0,
307         "value": [
308             {
309                 "key": 0,
310                 "value": true
311             },
312             {
313                 "key": "ALL_OTHERS",
314                 "value": false
315             }
316         ]
317     },
318     {
319         "key": "ALL_OTHERS",
320         "value": [

```

```

321         {
322             "key": "ALL_OTHERS",
323             "value": false
324         }
325     ]
326 }
327 ],
328 "_operators": [
329     {
330         "key": 4,
331         "value": 8
332     },
333     {
334         "key": 32,
335         "value": 0
336     },
337     {
338         "key": 64,
339         "value": 0
340     },
341     {
342         "key": 2,
343         "value": 8
344     },
345     {
346         "key": 0,
347         "value": 4
348     },
349     {
350         "key": "ALL_OTHERS",
351         "value": 2
352     }
353 ],
354 "_metaTransactionContracts": [
355     {
356         "key": 32,
357         "value": true
358     },
359     {
360         "key": "ALL_OTHERS",
361         "value": false
362     }
363 ],
364 "_admin": 0,
365 "_superOperators": [
366     {
367         "key": "ALL_OTHERS",
368         "value": false
369     }
370 ]
371 }
372 }
373 },
374 {
375     "key": "ALL_OTHERS",
376     "value": "EmptyAddress"
377 }
378 ]

```

Formal Verification Request 300

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

10, Dec 2019

2.36 ms

Line 632 in File LandBaseToken.sol

632 `//@CTK NO_BUF_OVERFLOW`

Line 634-665 in File LandBaseToken.sol

```

634  function _regroup24x24(address from, address to, uint256 x, uint256 y, bool set)
        internal returns (bool) {
635      uint256 id = x + y * GRID_SIZE;
636      uint256 quadId = LAYER_24x24 + id;
637      bool ownerOfAll = true;
638      for (uint256 xi = x; xi < x+24; xi += 12) {
639          for (uint256 yi = y; yi < y+24; yi += 12) {
640              bool ownAllIndividual = _regroup12x12(from, to, xi, yi, false);
641              uint256 id12x12 = LAYER_12x12 + xi + yi * GRID_SIZE;
642              uint256 owner12x12 = _owners[id12x12];
643              if (owner12x12 != 0) {
644                  if(!ownAllIndividual) {
645                      require(owner12x12 == uint256(from), "not owner of 12x12 quad");
646                  }
647                  _owners[id12x12] = 0;
648              }
649              ownerOfAll = (ownAllIndividual || owner12x12 != 0) && ownerOfAll;
650          }
651      }
652      if(set) {
653          if(!ownerOfAll) {
654              require(
655                  _owners[quadId] == uint256(from),
656                  "not owner of all sub quads not parent quad"
657              );
658          }
659          _owners[quadId] = uint256(to);
660          return true;
661      }
662      return ownerOfAll || _owners[quadId] == uint256(from);
663  }

```

✓ The code meets the specification.

Formal Verification Request 301

Method will not encounter an assertion failure.

10, Dec 2019

1.36 ms

Line 633 in File LandBaseToken.sol

633 `//@CTK NO_ASF`

Line 634-665 in File LandBaseToken.sol

```

634     function _regroup24x24(address from, address to, uint256 x, uint256 y, bool set)
        internal returns (bool) {
635         uint256 id = x + y * GRID_SIZE;
636         uint256 quadId = LAYER_24x24 + id;
637         bool ownerOfAll = true;
638         for (uint256 xi = x; xi < x+24; xi += 12) {
639             for (uint256 yi = y; yi < y+24; yi += 12) {
640                 bool ownAllIndividual = _regroup12x12(from, to, xi, yi, false);
641                 uint256 id12x12 = LAYER_12x12 + xi + yi * GRID_SIZE;
642                 uint256 owner12x12 = _owners[id12x12];
643                 if (owner12x12 != 0) {
644                     if(!ownAllIndividual) {
645                         require(owner12x12 == uint256(from), "not owner of 12x12 quad");
646                     }
647                     _owners[id12x12] = 0;
648                 }
649                 ownerOfAll = (ownAllIndividual || owner12x12 != 0) && ownerOfAll;
650             }
651         }
652         if(set) {
653             if(!ownerOfAll) {
654                 require(
655                     _owners[quadId] == uint256(from),
656                     "not owner of all sub quads not parent quad"
657                 );
658             }
659             _owners[quadId] = uint256(to);
660             return true;
661         }
662         return ownerOfAll || _owners[quadId] == uint256(from);
663     }

```

✓ The code meets the specification.

Formal Verification Request 302

If method completes, integer overflow would not happen.

📅 10, Dec 2019

🕒 0.84 ms

Line 667 in File LandBaseToken.sol

```

667     //@CTK NO_OVERFLOW

```

Line 675-701 in File LandBaseToken.sol

```

675     function _ownerOf(uint256 id) internal view returns (address) {
676         require(id & LAYER == 0, "Invalid token id");
677         uint256 x = id % GRID_SIZE;
678         uint256 y = id / GRID_SIZE;
679         uint256 owner1x1 = _owners[id];
680
681         if (owner1x1 != 0) {
682             return address(owner1x1); // cast to zero
683         } else {

```

```

684     address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) *
        GRID_SIZE]);
685     if (owner3x3 != address(0)) {
686         return owner3x3;
687     } else {
688         address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) *
        GRID_SIZE]);
689         if (owner6x6 != address(0)) {
690             return owner6x6;
691         } else {
692             address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) *
        12) * GRID_SIZE]);
693             if (owner12x12 != address(0)) {
694                 return owner12x12;
695             } else {
696                 return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) *
        GRID_SIZE]);
697             }
698         }
699     }
700 }
701 }

```

✓ The code meets the specification.

Formal Verification Request 303

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



10, Dec 2019



0.91 ms

Line 668 in File LandBaseToken.sol

```
668 //CTK NO_BUF_OVERFLOW
```

Line 675-701 in File LandBaseToken.sol

```

675 function _ownerOf(uint256 id) internal view returns (address) {
676     require(id & LAYER == 0, "Invalid token id");
677     uint256 x = id % GRID_SIZE;
678     uint256 y = id / GRID_SIZE;
679     uint256 owner1x1 = _owners[id];
680
681     if (owner1x1 != 0) {
682         return address(owner1x1); // cast to zero
683     } else {
684         address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) *
        GRID_SIZE]);
685         if (owner3x3 != address(0)) {
686             return owner3x3;
687         } else {
688             address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) *
        GRID_SIZE]);
689             if (owner6x6 != address(0)) {
690                 return owner6x6;
691             } else {

```

```

692         address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) *
693             12) * GRID_SIZE]);
694         if (owner12x12 != address(0)) {
695             return owner12x12;
696         } else {
697             return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) *
698                 GRID_SIZE]);
699         }
700     }
701 }

```

✓ The code meets the specification.

Formal Verification Request 304

Method will not encounter an assertion failure.



10, Dec 2019



1.13 ms

Line 669 in File LandBaseToken.sol

```
669 // @CTK NO_ASF
```

Line 675-701 in File LandBaseToken.sol

```

675 function _ownerOf(uint256 id) internal view returns (address) {
676     require(id & LAYER == 0, "Invalid token id");
677     uint256 x = id % GRID_SIZE;
678     uint256 y = id / GRID_SIZE;
679     uint256 owner1x1 = _owners[id];
680
681     if (owner1x1 != 0) {
682         return address(owner1x1); // cast to zero
683     } else {
684         address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) *
685             GRID_SIZE]);
686         if (owner3x3 != address(0)) {
687             return owner3x3;
688         } else {
689             address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) *
690                 GRID_SIZE]);
691             if (owner6x6 != address(0)) {
692                 return owner6x6;
693             } else {
694                 address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) *
695                     12) * GRID_SIZE]);
696                 if (owner12x12 != address(0)) {
697                     return owner12x12;
698                 } else {
699                     return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) *
700                         GRID_SIZE]);
701                 }
702             }
703         }
704     }
705 }

```


701 }

✓ The code meets the specification.

Formal Verification Request 305

`__ownerOf`

10, Dec 2019



15.09 ms

Line 670-674 in File LandBaseToken.sol

```

670  /*@CTK FAIL "__ownerOf"
671     @pre GRID_SIZE == 408
672     @pre (id & LAYER) == 0
673     @post (_owners[id] != 0) -> (__return == address(_owners[id]))
674  */

```

Line 675-701 in File LandBaseToken.sol

```

675  function __ownerOf(uint256 id) internal view returns (address) {
676      require(id & LAYER == 0, "Invalid token id");
677      uint256 x = id % GRID_SIZE;
678      uint256 y = id / GRID_SIZE;
679      uint256 owner1x1 = _owners[id];
680
681      if (owner1x1 != 0) {
682          return address(owner1x1); // cast to zero
683      } else {
684          address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) *
              GRID_SIZE]);
685          if (owner3x3 != address(0)) {
686              return owner3x3;
687          } else {
688              address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) *
              GRID_SIZE]);
689              if (owner6x6 != address(0)) {
690                  return owner6x6;
691              } else {
692                  address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) *
              12) * GRID_SIZE]);
693                  if (owner12x12 != address(0)) {
694                      return owner12x12;
695                  } else {
696                      return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) *
              GRID_SIZE]);
697                  }
698              }
699          }
700      }
701  }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {

```

```

4      id = 0
5  }
6  This = 0
7  Internal = {
8      __has_assertion_failure = false
9      __has_buf_overflow = false
10     __has_overflow = false
11     __has_returned = false
12     __reverted = false
13     msg = {
14         "gas": 0,
15         "sender": 0,
16         "value": 0
17     }
18 }
19 Other = {
20     __return = 0
21     block = {
22         "number": 0,
23         "timestamp": 0
24     }
25 }
26 Address_Map = [
27     {
28         "key": "ALL_OTHERS",
29         "value": {
30             "contract_name": "LandBaseToken",
31             "balance": 0,
32             "contract": {
33                 "GRID_SIZE": 152,
34                 "LAYER": 0,
35                 "LAYER_1x1": 0,
36                 "LAYER_3x3": 0,
37                 "LAYER_6x6": 0,
38                 "LAYER_12x12": 0,
39                 "LAYER_24x24": 0,
40                 "_minters": [
41                     {
42                         "key": "ALL_OTHERS",
43                         "value": false
44                     }
45                 ],
46                 "_ERC721_RECEIVED": "AAAA",
47                 "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
48                 "ERC165ID": "AAAA",
49                 "ERC721_MANDATORY_RECEIVER": "AAAA",
50                 "_numNFTPerAddress": [
51                     {
52                         "key": 4,
53                         "value": 16
54                     },
55                     {
56                         "key": "ALL_OTHERS",
57                         "value": 0
58                     }
59                 ],
60                 "_owners": [
61                     {

```

```

62         "key": 80,
63         "value": 8
64     },
65     {
66         "key": 8,
67         "value": 64
68     },
69     {
70         "key": 0,
71         "value": 1
72     },
73     {
74         "key": 128,
75         "value": 2
76     },
77     {
78         "key": 64,
79         "value": 128
80     },
81     {
82         "key": "ALL_OTHERS",
83         "value": 32
84     }
85 ],
86 "_operatorsForAll": [
87     {
88         "key": "ALL_OTHERS",
89         "value": [
90             {
91                 "key": "ALL_OTHERS",
92                 "value": true
93             }
94         ]
95     }
96 ],
97 "_operators": [
98     {
99         "key": 80,
100        "value": 0
101    },
102    {
103        "key": 2,
104        "value": 0
105    },
106    {
107        "key": 0,
108        "value": 32
109    },
110    {
111        "key": "ALL_OTHERS",
112        "value": 8
113    }
114 ],
115 "_metaTransactionContracts": [
116     {
117         "key": 0,
118         "value": true
119     },

```

```
120     {
121         "key": "ALL_OTHERS",
122         "value": false
123     }
124 ],
125 "_admin": 0,
126 "_superOperators": [
127     {
128         "key": "ALL_OTHERS",
129         "value": true
130     }
131 ]
132 }
133 }
134 }
135 ]
136
137 Function invocation is reverted.
```

Formal Verification Request 306

`__checkBatchReceiverAcceptQuad__forloop__Generated`



10, Dec 2019



162.87 ms

(Loop) Line 347-353 in File LandBaseToken.sol

```
347     /*@CTK __checkBatchReceiverAcceptQuad_forloop
348         @inv i <= size * size
349         @pre size >= 1
350         @pre GRID_SIZE == 408
351         @post i == size * size
352         @post !__should_return
353     */
```

(Loop) Line 347-356 in File LandBaseToken.sol

```
347     /*@CTK __checkBatchReceiverAcceptQuad_forloop
348         @inv i <= size * size
349         @pre size >= 1
350         @pre GRID_SIZE == 408
351         @post i == size * size
352         @post !__should_return
353     */
354     for (uint256 i = 0; i < size*size; i++) {
355         ids[i] = _idInPath(i, size, x, y);
356     }
```

✓ The code meets the specification.

Formal Verification Request 307

`__transferQuad__loop__Generated`



10, Dec 2019



36.88 ms

(Loop) Line 454-457 in File LandBaseToken.sol

```
454      /*@CTK _transferQuad_loop
455         @inv i <= size * size
456         @post i == size * size
457      */
```

(Loop) Line 454-460 in File LandBaseToken.sol

```
454      /*@CTK _transferQuad_loop
455         @inv i <= size * size
456         @post i == size * size
457      */
458      for (uint256 i = 0; i < size*size; i++) {
459          emit Transfer(from, to, _idInPath(i, size, x, y));
460      }
```

✔ The code meets the specification.

Source Code with CertiK Labels

File LandSale.sol

```

1  pragma solidity 0.5.9;
2
3  import "../sandbox-private-contracts/src/Land.sol";
4  import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC20.sol";
5  import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/
    MetaTransactionReceiver.sol";
6
7
8  /**
9   * @title Land Sale contract
10  * @notice This contract manages the sale of our lands
11  */
12  contract LandSale is MetaTransactionReceiver {
13
14      uint256 internal constant GRID_SIZE = 408; // 408 is the size of the Land
15
16      Land internal _land;
17      ERC20 internal _sand;
18      address payable internal _wallet;
19      uint256 internal _expiryTime;
20      bytes32 internal _merkleRoot;
21
22      event LandQuadPurchased(
23          address indexed buyer,
24          address indexed to,
25          uint256 indexed topCornerId,
26          uint256 size,
27          uint256 price
28      );
29
30      //@CTK NO_OVERFLOW
31      //@CTK NO_BUF_OVERFLOW
32      //@CTK NO_ASF
33      /*CTK LandSale
34          @tag assume_completion
35          @post __post._land == landAddress
36          @post __post._sand == sandContractAddress
37          @post __post._metaTransactionContracts[initialMetaTx] == true
38          @post __post._admin == admin
39          @post __post._wallet == initialWalletAddress
40          @post __post._merkleRoot == merkleRoot
41          @post __post._expiryTime == expiryTime
42      */
43      constructor(
44          address landAddress,
45          address sandContractAddress,
46          address initialMetaTx,
47          address admin,
48          address payable initialWalletAddress,
49          bytes32 merkleRoot,
50          uint256 expiryTime
51      ) public {
52          _land = Land(landAddress);
53          _sand = ERC20(sandContractAddress);

```

```

54     _setMetaTransactionProcessor(initialMetaTx, true);
55     _admin = admin;
56     _wallet = initialWalletAddress;
57     _merkleRoot = merkleRoot;
58     _expiryTime = expiryTime;
59 }
60
61 /// @notice set the wallet receiving the proceeds
62 /// @param newWallet address of the new receiving wallet
63 ///@CTK NO_OVERFLOW
64 ///@CTK NO_BUF_OVERFLOW
65 ///@CTK NO_ASF
66 /*@CTK setReceivingWallet_require
67     @tag assume_completion
68     @post newWallet != address(0)
69     @post msg.sender == _admin
70 */
71 /*@CTK setReceivingWallet_change
72     @tag assume_completion
73     @post __post._wallet == newWallet
74 */
75 function setReceivingWallet(address payable newWallet) external{
76     require(newWallet != address(0), "receiving wallet cannot be zero address");
77     require(msg.sender == _admin, "only admin can change the receiving wallet");
78     _wallet = newWallet;
79 }
80
81 /**
82  * @notice buy Land using the merkle proof associated with it
83  * @param buyer address that perform the payment
84  * @param to address that will own the purchased Land
85  * @param reserved the reserved address (if any)
86  * @param x x coordinate of the Land
87  * @param y y coordinate of the Land
88  * @param size size of the pack of Land to purchase
89  * @param price amount of Sand to purchase that Land
90  * @param proof merkleProof for that particular Land
91  * @return The address of the operator
92  */
93 ///@CTK NO_OVERFLOW
94 ///@CTK NO_BUF_OVERFLOW
95 ///@CTK NO_ASF
96 /*@CTK buyLandWithSand
97     @tag assume_completion
98     @pre _expiryTime > block.timestamp
99     @post buyer == msg.sender \\/ _metaTransactionContracts[msg.sender] == true
100     @post reserved == address(0) \\/ reserved == buyer
101 */
102 function buyLandWithSand(
103     address buyer,
104     address to,
105     address reserved,
106     uint256 x,
107     uint256 y,
108     uint256 size,
109     uint256 price,
110     bytes32 salt,
111     bytes32[] calldata proof

```

```

112 ) external {
113     /* solhint-disable-next-line not-rely-on-time */
114     require(block.timestamp < _expiryTime, "sale is over");
115     require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
        authorized");
116     require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
117     bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
118
119     require(
120         _verify(proof, leaf),
121         "Invalid land provided"
122     );
123
124     require(
125         _sand.transferFrom(
126             buyer,
127             _wallet,
128             price
129         ),
130         "sand transfer failed"
131     );
132
133     _land.mintQuad(to, size, x, y, "");
134     emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);
135 }
136
137 /**
138  * @notice Gets the expiry time for the current sale
139  * @return The expiry time, as a unix epoch
140  */
141 /*@CTK getExpiryTime
142  @post __return == _expiryTime
143  */
144 function getExpiryTime() external view returns(uint256) {
145     return _expiryTime;
146 }
147
148 /**
149  * @notice Gets the Merkle root associated with the current sale
150  * @return The Merkle root, as a bytes32 hash
151  */
152 /*@CTK merkleRoot
153  @post __return == _merkleRoot
154  */
155 function merkleRoot() external view returns(bytes32) {
156     return _merkleRoot;
157 }
158
159 //@CTK NO_OVERFLOW
160 //@CTK NO_BUF_OVERFLOW
161 //@CTK NO_ASF
162 function _generateLandHash(
163     uint256 x,
164     uint256 y,
165     uint256 size,
166     uint256 price,
167     address reserved,
168     bytes32 salt

```



```

169     ) internal pure returns (
170         bytes32
171     ) {
172         return keccak256(
173             abi.encodePacked(
174                 x,
175                 y,
176                 size,
177                 price,
178                 reserved,
179                 salt
180             )
181         );
182     }
183
184     //@CTK NO_OVERFLOW
185     //@CTK NO_BUF_OVERFLOW
186     //@CTK NO_ASF
187     function _verify(bytes32[] memory proof, bytes32 leaf) internal view returns (bool) {
188         bytes32 computedHash = leaf;
189
190         /*@CTK _verify_loop
191          @inv i <= proof.length
192          @post i == proof.length
193          */
194         for (uint256 i = 0; i < proof.length; i++) {
195             bytes32 proofElement = proof[i];
196
197             if (computedHash < proofElement) {
198                 computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
199             } else {
200                 computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
201             }
202         }
203
204         return computedHash == _merkleRoot;
205     }
206 }

```

File SafeMathWithRequire.sol

```

1  pragma solidity ^0.5.2;
2
3  /**
4   * @title SafeMath
5   * @dev Math operations with safety checks that revert
6   */
7  library SafeMathWithRequire {
8      /**
9       * @dev Multiplies two numbers, throws on overflow.
10      */
11      //@CTK NO_OVERFLOW
12      //@CTK NO_BUF_OVERFLOW
13      //@CTK NO_ASF
14      /*@CTK mul
15       @tag assume_completion
16       @post __return == a * b
17       */
18      function mul(uint256 a, uint256 b) internal pure returns (uint256) {

```

```

19      // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
20      // benefit is lost if 'b' is also tested.
21      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
22      if (a == 0) {
23          return 0;
24      }
25
26      uint256 c = a * b;
27      require(c / a == b, "overflow");
28      return c;
29  }
30
31  /**
32   * @dev Integer division of two numbers, truncating the quotient.
33   */
34  //@CTK NO_OVERFLOW
35  //@CTK NO_BUF_OVERFLOW
36  //@CTK FAIL NO_ASF
37  /*CTK div
38   @tag assume_completion
39   @post __return == a / b
40  */
41  function div(uint256 a, uint256 b) internal pure returns (uint256) {
42      // assert(b > 0); // Solidity automatically throws when dividing by 0
43      // uint256 c = a / b;
44      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45      return a / b;
46  }
47
48  /**
49   * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than
50       minuend).
51   */
52  //@CTK NO_OVERFLOW
53  //@CTK NO_BUF_OVERFLOW
54  //@CTK NO_ASF
55  /*CTK sub
56   @tag assume_completion
57   @post __return == a - b
58  */
59  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
60      require(b <= a, "undeflow");
61      return a - b;
62  }
63
64  /**
65   * @dev Adds two numbers, throws on overflow.
66   */
67  //@CTK NO_OVERFLOW
68  //@CTK NO_BUF_OVERFLOW
69  //@CTK NO_ASF
70  /*CTK add
71   @tag assume_completion
72   @post __return == a + b
73  */
74  function add(uint256 a, uint256 b) internal pure returns (uint256) {
75      uint256 c = a + b;
76      require(c >= a, "overflow");

```

```
76     return c;
77 }
78 }
```

File LandSaleWithETHAndDAI.sol

```
1  pragma solidity 0.5.9;
2
3  import "../sandbox-private-contracts/contracts_common/src/Libraries/SafeMathWithRequire.sol";
4  import "../sandbox-private-contracts/src/Land.sol";
5  import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC20.sol";
6  import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/
    MetaTransactionReceiver.sol";
7  import "../sandbox-private-contracts/contracts_common/src/Interfaces/Medianizer.sol";
8
9  /**
10   * @title Land Sale contract that supports also DAI and ETH as payment
11   * @notice This contract manages the sale of our lands
12   */
13  contract LandSaleWithETHAndDAI is MetaTransactionReceiver {
14      using SafeMathWithRequire for uint256;
15
16      uint256 internal constant GRID_SIZE = 408; // 408 is the size of the Land
17      uint256 internal constant daiPrice = 1440000000000000000;
18
19      Land internal _land;
20      ERC20 internal _sand;
21      Medianizer private _medianizer;
22      ERC20 private _dai;
23
24      address payable internal _wallet;
25      uint256 internal _expiryTime;
26      bytes32 internal _merkleRoot;
27
28      bool _sandEnabled = true;
29      bool _etherEnabled = false;
30      bool _daiEnabled = false;
31
32      event LandQuadPurchased(
33          address indexed buyer,
34          address indexed to,
35          uint256 indexed topCornerId,
36          uint256 size,
37          uint256 price,
38          address token,
39          uint256 amountPaid
40      );
41
42      // @CTK NO_OVERFLOW
43      // @CTK NO_BUF_OVERFLOW
44      // @CTK NO_ASF
45      /* @CTK LandSale
46         @tag assume_completion
47         @post __post._land == landAddress
48         @post __post._sand == sandContractAddress
49         @post __post._metaTransactionContracts[initialMetaTx] == true
50         @post __post._admin == admin
51         @post __post._wallet == initialWalletAddress
```

```

52     @post __post._merkleRoot == merkleRoot
53     @post __post._expiryTime == expiryTime
54     @post __post._medianizer == medianizerContractAddress
55     @post __post._dai == daiTokenContractAddress
56     */
57     constructor(
58         address landAddress,
59         address sandContractAddress,
60         address initialMetaTx,
61         address admin,
62         address payable initialWalletAddress,
63         bytes32 merkleRoot,
64         uint256 expiryTime,
65         address medianizerContractAddress,
66         address daiTokenContractAddress
67     ) public {
68         _land = Land(landAddress);
69         _sand = ERC20(sandContractAddress);
70         _setMetaTransactionProcessor(initialMetaTx, true);
71         _admin = admin;
72         _wallet = initialWalletAddress;
73         _merkleRoot = merkleRoot;
74         _expiryTime = expiryTime;
75         _medianizer = Medianizer(medianizerContractAddress);
76         _dai = ERC20(daiTokenContractAddress);
77     }
78
79     /// @notice set the wallet receiving the proceeds
80     /// @param newWallet address of the new receiving wallet
81     ///@CTK NO_OVERFLOW
82     ///@CTK NO_BUF_OVERFLOW
83     ///@CTK NO_ASF
84     /*@CTK setReceivingWallet_require
85         @tag assume_completion
86         @post newWallet != address(0)
87         @post msg.sender == _admin
88     */
89     /*@CTK setReceivingWallet_change
90         @tag assume_completion
91         @post __post._wallet == newWallet
92     */
93     function setReceivingWallet(address payable newWallet) external{
94         require(newWallet != address(0), "receiving wallet cannot be zero address");
95         require(msg.sender == _admin, "only admin can change the receiving wallet");
96         _wallet = newWallet;
97     }
98
99     /// @notice enable/disable DAI payment for Lands
100    /// @param enabled whether to enable or disable
101    ///@CTK NO_OVERFLOW
102    ///@CTK NO_BUF_OVERFLOW
103    ///@CTK NO_ASF
104    /*@CTK setDAIEnabled_require
105        @tag assume_completion
106        @post msg.sender == _admin
107    */
108    /*@CTK setDAIEnabled_change
109        @tag assume_completion

```

```

110     @post __post._daiEnabled == enabled
111     */
112     function setDAIEnabled(bool enabled) external {
113         require(msg.sender == _admin, "only admin can enable/disable DAI");
114         _daiEnabled = enabled;
115     }
116
117     /// @notice return whether DAI payments are enabled
118     /// @return whether DAI payments are enabled
119     ///@CTK NO_OVERFLOW
120     ///@CTK NO_BUF_OVERFLOW
121     ///@CTK NO_ASF
122     /*@CTK isDAIEnabled
123     @post __return == _daiEnabled
124     */
125     function isDAIEnabled() external view returns (bool) {
126         return _daiEnabled;
127     }
128
129     /// @notice enable/disable ETH payment for Lands
130     /// @param enabled whether to enable or disable
131     ///@CTK NO_OVERFLOW
132     ///@CTK NO_BUF_OVERFLOW
133     ///@CTK NO_ASF
134     /*@CTK setETHEnabled_require
135     @tag assume_completion
136     @post msg.sender == _admin
137     */
138     /*@CTK setETHEnabled_change
139     @tag assume_completion
140     @post __post._etherEnabled == enabled
141     */
142     function setETHEnabled(bool enabled) external {
143         require(msg.sender == _admin, "only admin can enable/disable ETH");
144         _etherEnabled = enabled;
145     }
146
147     /// @notice return whether ETH payments are enabled
148     /// @return whether ETH payments are enabled
149     ///@CTK NO_OVERFLOW
150     ///@CTK NO_BUF_OVERFLOW
151     ///@CTK NO_ASF
152     /*@CTK isETHEnabled
153     @post __return == _etherEnabled
154     */
155     function isETHEnabled() external view returns (bool) {
156         return _etherEnabled;
157     }
158
159     /// @notice enable/disable the specific SAND payment for Lands
160     /// @param enabled whether to enable or disable
161     ///@CTK NO_OVERFLOW
162     ///@CTK NO_BUF_OVERFLOW
163     ///@CTK NO_ASF
164     /*@CTK setSANDEnabled_require
165     @tag assume_completion
166     @post msg.sender == _admin
167     */

```

```

168  /*@CTK setSANDEnabled_change
169      @tag assume_completion
170      @post __post._sandEnabled == enabled
171  */
172  function setSANDEnabled(bool enabled) external {
173      require(msg.sender == _admin, "only admin can enable/disable SAND");
174      _sandEnabled = enabled;
175  }
176
177  /// @notice return whether the specific SAND payments are enabled
178  /// @return whether the specific SAND payments are enabled
179  //@CTK NO_OVERFLOW
180  //@CTK NO_BUF_OVERFLOW
181  //@CTK NO_ASF
182  /*@CTK isSANDEnabled
183      @post __return == _sandEnabled
184  */
185  function isSANDEnabled() external view returns (bool) {
186      return _sandEnabled;
187  }
188
189  //@CTK NO_OVERFLOW
190  //@CTK NO_BUF_OVERFLOW
191  //@CTK NO_ASF
192  /*@CTK _checkValidity
193      @tag assume_completion
194      @post buyer == msg.sender \\/ _metaTransactionContracts[msg.sender] == true
195      @post reserved == address(0) \\/ reserved == buyer
196  */
197  function _checkValidity(
198      address buyer,
199      address reserved,
200      uint256 x,
201      uint256 y,
202      uint256 size,
203      uint256 price,
204      bytes32 salt,
205      bytes32[] memory proof
206  ) internal view {
207      /* solium-disable-next-line security/no-block-members */
208      require(block.timestamp < _expiryTime, "sale is over");
209      require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
        authorized");
210      require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
211      bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);
212
213      require(
214          _verify(proof, leaf),
215          "Invalid land provided"
216      );
217  }
218
219  //@FIXME NO_OVERFLOW
220  //@FIXME NO_BUF_OVERFLOW
221  //@FIXME NO_ASF
222  function _mint(address buyer, address to, uint256 x, uint256 y, uint256 size, uint256
        price, address token, uint256 tokenAmount) internal {
223      _land.mintQuad(to, size, x, y, "");

```

```

224     emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price, token,
225         tokenAmount);
226 }
227 /**
228  * @notice buy Land with SAND using the merkle proof associated with it
229  * @param buyer address that perform the payment
230  * @param to address that will own the purchased Land
231  * @param reserved the reserved address (if any)
232  * @param x x coordinate of the Land
233  * @param y y coordinate of the Land
234  * @param size size of the pack of Land to purchase
235  * @param priceInSand price in SAND to purchase that Land
236  * @param proof merkleProof for that particular Land
237  * @return The address of the operator
238  */
239 //CTK NO_OVERFLOW
240 //CTK NO_BUF_OVERFLOW
241 //CTK NO_ASF
242 /*CTK buyLandWithSand
243  @tag assume_completion
244  @post _sandEnabled == true
245  @post buyer == msg.sender /\ _metaTransactionContracts[msg.sender] == true
246  @post reserved == address(0) /\ reserved == buyer
247  */
248 function buyLandWithSand(
249     address buyer,
250     address to,
251     address reserved,
252     uint256 x,
253     uint256 y,
254     uint256 size,
255     uint256 priceInSand,
256     bytes32 salt,
257     bytes32[] calldata proof
258 ) external {
259     require(!_sandEnabled, "sand payments not enabled");
260     _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
261     require(
262         _sand.transferFrom(
263             buyer,
264             _wallet,
265             priceInSand
266         ),
267         "sand token transfer failed"
268     );
269     _mint(buyer, to, x, y, size, priceInSand, address(_sand), priceInSand);
270 }
271 /**
272  * @notice buy Land with ETH using the merkle proof associated with it
273  * @param buyer address that perform the payment
274  * @param to address that will own the purchased Land
275  * @param reserved the reserved address (if any)
276  * @param x x coordinate of the Land
277  * @param y y coordinate of the Land
278  * @param size size of the pack of Land to purchase
279  * @param priceInSand price in SAND to purchase that Land
280  */

```

```

281     * @param proof merkleProof for that particular Land
282     * @return The address of the operator
283     */
284     //@CTK NO_OVERFLOW
285     //@CTK NO_BUF_OVERFLOW
286     //@CTK FAIL NO_ASF
287     /*@CTK buyLandWithETH_require
288         @tag assume_completion
289         @post _sandEnabled == true
290         @post buyer == msg.sender /\ _metaTransactionContracts[msg.sender] == true
291         @post reserved == address(0) /\ reserved == buyer
292     */
293     function buyLandWithETH(
294         address buyer,
295         address to,
296         address reserved,
297         uint256 x,
298         uint256 y,
299         uint256 size,
300         uint256 priceInSand,
301         bytes32 salt,
302         bytes32[] calldata proof
303     ) external payable {
304         require(_etherEnabled, "ether payments not enabled");
305         _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
306
307         uint256 ETHRequired = getEtherAmountWithSAND(priceInSand);
308         require(msg.value >= ETHRequired, "not enough ether sent");
309         uint256 leftOver = msg.value - ETHRequired;
310         if(leftOver > 0) {
311             msg.sender.transfer(leftOver); // refund extra
312         }
313         address(_wallet).transfer(ETHRequired);
314
315         _mint(buyer, to, x, y, size, priceInSand, address(0), ETHRequired);
316     }
317
318     /**
319     * @notice buy Land with DAI using the merkle proof associated with it
320     * @param buyer address that perform the payment
321     * @param to address that will own the purchased Land
322     * @param reserved the reserved address (if any)
323     * @param x x coordinate of the Land
324     * @param y y coordinate of the Land
325     * @param size size of the pack of Land to purchase
326     * @param priceInSand price in SAND to purchase that Land
327     * @param proof merkleProof for that particular Land
328     * @return The address of the operator
329     */
330     //@CTK NO_OVERFLOW
331     //@CTK NO_BUF_OVERFLOW
332     //@CTK FAIL NO_ASF
333     /*@CTK buyLandWithDAI
334         @tag assume_completion
335         @post _sandEnabled == true
336         @post buyer == msg.sender /\ _metaTransactionContracts[msg.sender] == true
337         @post reserved == address(0) /\ reserved == buyer
338     */

```



```

339 function buyLandWithDAI(
340     address buyer,
341     address to,
342     address reserved,
343     uint256 x,
344     uint256 y,
345     uint256 size,
346     uint256 priceInSand,
347     bytes32 salt,
348     bytes32[] calldata proof
349 ) external {
350     require(!_daiEnabled, "dai payments not enabled");
351     _checkValidity(buyer, reserved, x, y, size, priceInSand, salt, proof);
352
353     uint256 DAIRequired = priceInSand.mul(daiPrice).div(1000000000000000000);
354     require(!_dai.transferFrom(msg.sender, _wallet, DAIRequired), "failed to transfer dai
        ");
355
356     _mint(buyer, to, x, y, size, priceInSand, address(_dai), DAIRequired);
357 }
358
359 /**
360  * @notice Gets the expiry time for the current sale
361  * @return The expiry time, as a unix epoch
362  */
363 /*@CTK getExpiryTime
364  @post __return == _expiryTime
365  */
366 function getExpiryTime() external view returns(uint256) {
367     return _expiryTime;
368 }
369
370 /**
371  * @notice Gets the Merkle root associated with the current sale
372  * @return The Merkle root, as a bytes32 hash
373  */
374 /*@CTK merkleRoot
375  @post __return == _merkleRoot
376  */
377 function merkleRoot() external view returns(bytes32) {
378     return _merkleRoot;
379 }
380
381 /*@CTK NO_OVERFLOW
382 /*@CTK NO_BUF_OVERFLOW
383 /*@CTK NO_ASF
384 function _generateLandHash(
385     uint256 x,
386     uint256 y,
387     uint256 size,
388     uint256 price,
389     address reserved,
390     bytes32 salt
391 ) internal pure returns (
392     bytes32
393 ) {
394     return keccak256(
395         abi.encodePacked(

```

```

396         x,
397         y,
398         size,
399         price,
400         reserved,
401         salt
402     )
403 );
404 }
405
406 function _verify(bytes32[] memory proof, bytes32 leaf) internal view returns (bool) {
407     bytes32 computedHash = leaf;
408
409     for (uint256 i = 0; i < proof.length; i++) {
410         bytes32 proofElement = proof[i];
411
412         if (computedHash < proofElement) {
413             computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
414         } else {
415             computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
416         }
417     }
418
419     return computedHash == _merkleRoot;
420 }
421
422 /**
423  * @notice Returns the amount of ETH for a specific amount of SAND
424  * @param sandAmount An amount of SAND
425  * @return The amount of ETH
426  */
427 // @CTK NO_OVERFLOW
428 // @CTK NO_BUF_OVERFLOW
429 // @CTK FAIL NO_ASF
430 function getEtherAmountWithSAND(uint256 sandAmount) public view returns (uint256) {
431     uint256 ethUsdPair = getEthUsdPair();
432     return sandAmount.mul(daiPrice).div(ethUsdPair);
433 }
434
435 /**
436  * @notice Gets the ETHUSD pair from the Medianizer contract
437  * @return The pair as an uint256
438  */
439 function getEthUsdPair() internal view returns (uint256) {
440     bytes32 pair = _medianizer.read();
441     return uint256(pair);
442 }
443 }

```

File Admin.sol

```

1 pragma solidity ^0.5.2;
2
3 contract Admin {
4
5     address internal _admin;
6
7     event AdminChanged(address oldAdmin, address newAdmin);
8

```

```

9      /// @notice gives the current administrator of this contract.
10     /// @return the current administrator of this contract.
11     /*@CTK getAdmin
12     @post __return == _admin
13     */
14     function getAdmin() external view returns (address) {
15         return _admin;
16     }
17
18     /// @notice change the administrator to be `newAdmin`.
19     /// @param newAdmin address of the new administrator.
20     /*@CTK NO_OVERFLOW
21     /*@CTK NO_BUF_OVERFLOW
22     /*@CTK NO_ASF
23     /*@CTK changeAdmin_requirement
24     @tag assume_completion
25     @post msg.sender == _admin
26     */
27     /*@CTK changeAdmin_change
28     @tag assume_completion
29     @pre msg.sender == _admin
30     @post __post._admin == newAdmin
31     */
32     function changeAdmin(address newAdmin) external {
33         require(msg.sender == _admin, "only admin can change admin");
34         emit AdminChanged(_admin, newAdmin);
35         _admin = newAdmin;
36     }
37
38     modifier onlyAdmin() {
39         require (msg.sender == _admin, "only admin allowed");
40         _;
41     }
42
43 }

```

File MetaTransactionReceiver.sol

```

1  pragma solidity ^0.5.2;
2
3  import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/Admin.sol";
4
5  contract MetaTransactionReceiver is Admin{
6
7      mapping(address => bool) internal _metaTransactionContracts;
8      event MetaTransactionProcessor(address metaTransactionProcessor, bool enabled);
9
10     /// @notice Enable or disable the ability of `metaTransactionProcessor` to perform meta-
11     tx (metaTransactionProcessor rights).
12     /// @param metaTransactionProcessor address that will be given/removed
13     metaTransactionProcessor rights.
14     /// @param enabled set whether the metaTransactionProcessor is enabled or disabled.
15     /*@CTK NO_OVERFLOW
16     /*@CTK NO_BUF_OVERFLOW
17     /*@CTK NO_ASF
18     /*@CTK setMetaTransactionProcessor
19     @tag assume_completion
20     @post msg.sender == _admin
21     */

```

```

20  /*@CTK setMetaTransactionProcessor
21      @tag assume_completion
22      @inv msg.sender == _admin
23      @post __post._metaTransactionContracts[metaTransactionProcessor] == enabled
24  */
25  function setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
26      public {
27      require(
28          msg.sender == _admin,
29          "only admin can setup metaTransactionProcessors"
30      );
31      _setMetaTransactionProcessor(metaTransactionProcessor, enabled);
32  }
33  // @CTK NO_OVERFLOW
34  // @CTK NO_BUF_OVERFLOW
35  // @CTK NO_ASF
36  /*@CTK _setMetaTransactionProcessor
37      @tag assume_completion
38      @post __post._metaTransactionContracts[metaTransactionProcessor] == enabled
39  */
40  function _setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
41      internal {
42      _metaTransactionContracts[metaTransactionProcessor] = enabled;
43      emit MetaTransactionProcessor(metaTransactionProcessor, enabled);
44  }
45  /// @notice check whether address `who` is given meta-transaction execution rights.
46  /// @param who The address to query.
47  /// @return whether the address has meta-transaction execution rights.
48  // @CTK NO_OVERFLOW
49  // @CTK NO_BUF_OVERFLOW
50  // @CTK NO_ASF
51  /*@CTK isMetaTransactionProcessor
52      @tag assume_completion
53      @post __return == _metaTransactionContracts[who]
54  */
55  function isMetaTransactionProcessor(address who) external view returns(bool) {
56      return _metaTransactionContracts[who];
57  }
58  }

```

File ERC721BaseToken.sol

```

1  /* solhint-disable func-order, code-complexity */
2  pragma solidity 0.5.9;
3
4  import "../sandbox-private-contracts/contracts_common/src/Libraries/AddressUtils.sol";
5  import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC721TokenReceiver.
6      sol";
7  import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC721Events.sol";
8  import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/SuperOperators.
9      sol";
10 import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/
11     MetaTransactionReceiver.sol";
12 import "../sandbox-private-contracts/contracts_common/src/Interfaces/
13     ERC721MandatoryTokenReceiver.sol";
14
15 contract ERC721BaseToken is ERC721Events, SuperOperators, MetaTransactionReceiver {

```

```

12  using AddressUtils for address;
13
14  bytes4 internal constant _ERC721_RECEIVED = 0x150b7a02;
15  bytes4 internal constant _ERC721_BATCH_RECEIVED = 0x4b808c46;
16
17  bytes4 internal constant ERC165ID = 0x01ffc9a7;
18  bytes4 internal constant ERC721_MANDATORY_RECEIVER = 0x5e8bf644;
19
20  mapping (address => uint256) public _numNFTPerAddress;
21  mapping (uint256 => uint256) public _owners;
22  mapping (address => mapping(address => bool)) public _operatorsForAll;
23  mapping (uint256 => address) public _operators;
24
25  //@CTK NO_OVERFLOW
26  //@CTK NO_BUF_OVERFLOW
27  //@CTK NO_ASF
28  /*@CTK ERC721BaseToken
29   @tag assume_completion
30   @post __post._admin == admin
31   @post __post._metaTransactionContracts[metaTransactionContract] == true
32  */
33  constructor(
34      address metaTransactionContract,
35      address admin
36  ) internal {
37      _admin = admin;
38      _setMetaTransactionProcessor(metaTransactionContract, true);
39  }
40
41  //@CTK FAIL NO_OVERFLOW
42  //@CTK NO_BUF_OVERFLOW
43  //@CTK NO_ASF
44  /*@CTK _transferFrom
45   @tag assume_completion
46   @pre from != to
47   @pre _numNFTPerAddress[from] > 0
48   @pre address(_owners[id]) == from
49   @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
50   @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
51   @post __post._owners[id] == uint256(to)
52  */
53  function _transferFrom(address from, address to, uint256 id) internal {
54      _numNFTPerAddress[from]--;
55      _numNFTPerAddress[to]++;
56      _owners[id] = uint256(to);
57      emit Transfer(from, to, id);
58  }
59
60  /**
61   * @notice Return the number of Land owned by an address
62   * @param owner The address to look for
63   * @return The number of Land token owned by the address
64   */
65  //@CTK NO_OVERFLOW
66  //@CTK NO_BUF_OVERFLOW
67  //@CTK NO_ASF
68  /*@CTK balanceOf_require
69   @tag assume_completion

```

```

70     @post owner != address(0)
71     */
72     /*@CTK balanceOf_change
73     @tag assume_completion
74     @pre owner != address(0)
75     @post __return == _numNFTPerAddress[owner]
76     */
77     function balanceOf(address owner) external view returns (uint256) {
78         require(owner != address(0), "owner is zero address");
79         return _numNFTPerAddress[owner];
80     }
81
82     //@CTK NO_OVERFLOW
83     //@CTK NO_BUF_OVERFLOW
84     /*@CTK _ownerOf
85     @post __return == address(_owners[id])
86     */
87     function _ownerOf(uint256 id) internal view returns (address) {
88         return address(_owners[id]);
89     }
90
91     //@CTK NO_OVERFLOW
92     //@CTK NO_BUF_OVERFLOW
93     //@CTK FAIL NO_ASF
94     /*@CTK _ownerAndOperatorEnabledOf
95     @post owner == address(_owners[id])
96     @post operatorEnabled == ((_owners[id] / 2**255) == 1)
97     */
98     function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner,
99         bool operatorEnabled) {
100         uint256 data = _owners[id];
101         owner = address(data);
102         operatorEnabled = (data / 2**255) == 1;
103     }
104
105     /**
106     * @notice Return the owner of a Land
107     * @param id The id of the Land
108     * @return The address of the owner
109     */
110     //@CTK NO_OVERFLOW
111     //@CTK NO_BUF_OVERFLOW
112     //@CTK NO_ASF
113     /*@CTK ownerOf
114     @tag assume_completion
115     @post owner == address(_owners[id])
116     @post owner != address(0)
117     */
118     function ownerOf(uint256 id) external view returns (address owner) {
119         owner = _ownerOf(id);
120         require(owner != address(0), "token does not exist");
121     }
122
123     //@CTK NO_OVERFLOW
124     //@CTK NO_BUF_OVERFLOW
125     //@CTK NO_ASF
126     /*@CTK _approveFor
127     @post (operator == address(0)) -> (__post._owners[id] == uint256(owner))

```

```

127     @post (operator != address(0)) -> (__post._owners[id] == uint256(owner) + 2**255)
128     @post (operator != address(0)) -> (__post._operators[id] == operator)
129     */
130     function _approveFor(address owner, address operator, uint256 id) internal {
131         if(operator == address(0)) {
132             _owners[id] = uint256(owner); // no need to reset the operator, it will be
133                 overridden next time
134         } else {
135             _owners[id] = uint256(owner) + 2**255;
136             _operators[id] = operator;
137         }
138         emit Approval(owner, operator, id);
139     }
140     /**
141     * @notice Approve an operator to spend tokens on the sender behalf
142     * @param sender The address giving the approval
143     * @param operator The address receiving the approval
144     * @param id The id of the token
145     */
146     //@CTK NO_OVERFLOW
147     //@CTK NO_BUF_OVERFLOW
148     //@CTK NO_ASF
149     /*CTK approveFor_require
150     @tag assume_completion
151     @post sender != address(0)
152     @post sender == address(_owners[id])
153     @post (msg.sender == sender) || (_metaTransactionContracts[msg.sender]) || (
154         _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])
155     */
156     /*CTK approveFor_change
157     @tag assume_completion
158     @pre sender != address(0)
159     @pre sender == address(_owners[id])
160     @pre (msg.sender == sender) || (_metaTransactionContracts[msg.sender]) || (
161         _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])
162     @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
163     @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
164         2**255)
165     @post (operator != address(0)) -> (__post._operators[id] == operator)
166     */
167     function approveFor(
168         address sender,
169         address operator,
170         uint256 id
171     ) external {
172         address owner = _ownerOf(id);
173         require(sender != address(0), "sender is zero address");
174         require(
175             msg.sender == sender ||
176             _metaTransactionContracts[msg.sender] ||
177             _superOperators[msg.sender] ||
178             _operatorsForAll[sender][msg.sender],
179             "not authorized to approve"
180         );
181         require(owner == sender, "owner != sender");
182         _approveFor(owner, operator, id);
183     }

```

```

181
182 /**
183  * @notice Approve an operator to spend tokens on the sender behalf
184  * @param operator The address receiving the approval
185  * @param id The id of the token
186  */
187 //@CTK NO_OVERFLOW
188 //@CTK NO_BUF_OVERFLOW
189 //@CTK NO_ASF
190 /*@CTK approve_require
191  @tag assume_completion
192  @post address(_owners[id]) != address(0)
193  @post (msg.sender == address(_owners[id])) || (_superOperators[msg.sender]) || (
    _operatorsForAll[address(_owners[id])][msg.sender])
194 */
195 /*@CTK approve_change
196  @tag assume_completion
197  @pre address(_owners[id]) != address(0)
198  @pre (msg.sender == address(_owners[id])) || (_superOperators[msg.sender]) || (
    _operatorsForAll[address(_owners[id])][msg.sender])
199  @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
200  @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
    2**255)
201  @post (operator != address(0)) -> (__post._operators[id] == operator)
202 */
203 function approve(address operator, uint256 id) external {
204     address owner = _ownerOf(id);
205     require(owner != address(0), "token does not exist");
206     require(
207         owner == msg.sender ||
208         _superOperators[msg.sender] ||
209         _operatorsForAll[owner][msg.sender],
210         "not authorized to approve"
211     );
212     _approveFor(owner, operator, id);
213 }
214
215 /**
216  * @notice Get the approved operator for a specific token
217  * @param id The id of the token
218  * @return The address of the operator
219  */
220 //@CTK NO_OVERFLOW
221 //@CTK NO_BUF_OVERFLOW
222 //@CTK FAIL NO_ASF
223 /*@CTK getApproved
224  @tag assume_completion
225  @post address(_owners[id]) != address(0)
226  @post ((__owners[id] / 2**255) == 1) -> (__return == _operators[id])
227  @post ((__owners[id] / 2**255) != 1) -> (__return == address(0))
228 */
229 function getApproved(uint256 id) external view returns (address) {
230     (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
231     require(owner != address(0), "token does not exist");
232     if (operatorEnabled) {
233         return _operators[id];
234     } else {
235         return address(0);

```



```

236     }
237 }
238
239 //@CTK NO_OVERFLOW
240 //@CTK NO_BUF_OVERFLOW
241 //@CTK FAIL NO_ASF
242 /*@CTK _checkTransfer
243   @tag assume_completion
244   @post address(_owners[id]) != address(0)
245   @post from == _owners[id]
246   @post to != address(0)
247   @post (msg.sender != from) && (_metaTransactionContracts[msg.sender] == false) ->
        _superOperators[msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id]
        ] / 2**255) == 1) && _operators[id] == msg.sender)
248 */
249 function _checkTransfer(address from, address to, uint256 id) internal view returns (
    bool isMetaTx) {
250     (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
251     require(owner != address(0), "token does not exist");
252     require(owner == from, "not owner in _checkTransfer");
253     require(to != address(0), "can't send to zero address");
254     isMetaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
255     if (msg.sender != from && !isMetaTx) {
256         require(
257             _superOperators[msg.sender] ||
258             _operatorsForAll[from][msg.sender] ||
259             (operatorEnabled && _operators[id] == msg.sender),
260             "not approved to transfer"
261         );
262     }
263 }
264
265 //@CTK NO_OVERFLOW
266 //@CTK NO_BUF_OVERFLOW
267 //@CTK NO_ASF
268 /*@CTK _checkInterfaceWith10000Gas
269   @tag assume_completion
270   @post __return == true
271 */
272 function _checkInterfaceWith10000Gas(address _contract, bytes4 interfaceId)
273     internal
274     view
275     returns (bool)
276 {
277     bool success;
278     bool result;
279     bytes memory call_data = abi.encodeWithSelector(
280         ERC165ID,
281         interfaceId
282     );
283     // solium-disable-next-line security/no-inline-assembly
284     /*@CTK _checkInterfaceWith10000Gas_assembly
285       @tag assume_completion
286       @var bool success
287       @var bool result
288       @post result == true
289       @post success == true
290     */

```

```

291 // solium-disable-next-line security/no-inline-assembly
292 assembly {
293     let call_ptr := add(0x20, call_data)
294     let call_size := mload(call_data)
295     let output := mload(0x40) // Find empty storage location using "free memory
                                // pointer"
296     mstore(output, 0x0)
297     success := staticcall(
298         10000,
299         _contract,
300         call_ptr,
301         call_size,
302         output,
303         0x20
304     ) // 32 bytes
305     result := mload(output)
306 }
307 // (10000 / 63) "not enough for supportsInterface(...)" // consume all gas, so
    // caller can potentially know that there was not enough gas
308 assert(gasleft() > 158);
309 return success && result;
310 }
311
312 /**
313  * @notice Transfer a token between 2 addresses
314  * @param from The sender of the token
315  * @param to The recipient of the token
316  * @param id The id of the token
317  */
318 //@CTK NO_OVERFLOW
319 //@CTK NO_BUF_OVERFLOW
320 //@CTK FAIL NO_ASF
321 /*@CTK transferFrom
322  @tag assume_completion
323  @pre (from == _owners[id]) && (from != address(0))
324  @pre to != 0
325  @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || _superOperators[
    msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id] / 2**255) ==
    1) && _operators[id] == msg.sender)
326  @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
327  @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
328  @post __post._owners[id] == uint256(to)
329  */
330 function transferFrom(address from, address to, uint256 id) external {
331     bool metaTx = _checkTransfer(from, to, id);
332     _transferFrom(from, to, id);
333     if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
334         require(
335             _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, ""),
336             "erc721 transfer rejected by to"
337         );
338     }
339 }
340
341 /**
342  * @notice Transfer a token between 2 addresses letting the receiver knows of the
    // transfer
343  * @param from The sender of the token

```

```

344     * @param to The recipient of the token
345     * @param id The id of the token
346     * @param data Additional data
347     */
348     //@CTK NO_OVERFLOW
349     //@CTK NO_BUF_OVERFLOW
350     //@CTK FAIL NO_ASF
351     /*@CTK safeTransferFrom
352     @tag assume_completion
353     @pre (from == _owners[id]) && (from != address(0))
354     @pre to != address(0)
355     @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || _superOperators[
        msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id] / 2**255) ==
        1) && _operators[id] == msg.sender)
356     @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
357     @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
358     @post __post._owners[id] == uint256(to)
359     */
360     function safeTransferFrom(address from, address to, uint256 id, bytes memory data)
        public {
361         bool metaTx = _checkTransfer(from, to, id);
362         _transferFrom(from, to, id);
363         if (to.isContract()) {
364             require(
365                 _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, data),
366                 "ERC721: transfer rejected by to"
367             );
368         }
369     }
370
371     /**
372     * @notice Transfer a token between 2 addresses letting the receiver knows of the
        transfer
373     * @param from The send of the token
374     * @param to The recipient of the token
375     * @param id The id of the token
376     */
377     function safeTransferFrom(address from, address to, uint256 id) external {
378         safeTransferFrom(from, to, id, "");
379     }
380
381     /**
382     * @notice Transfer many tokens between 2 addresses
383     * @param from The sender of the token
384     * @param to The recipient of the token
385     * @param ids The ids of the tokens
386     * @param data additional data
387     */
388     function batchTransferFrom(address from, address to, uint256[] calldata ids, bytes
        calldata data) external {
389         _batchTransferFrom(from, to, ids, data, false);
390     }
391
392     //@CTK FAIL NO_OVERFLOW
393     //@CTK NO_BUF_OVERFLOW
394     //@CTK NO_ASF
395     /*@CTK FAIL "_batchTransferFrom"
396     @tag assume_completion

```

```

397 @pre from != address(0)
398 @pre to != address(0)
399 @post _numNFTPerAddress[from] + _numNFTPerAddress[to] == (__post._numNFTPerAddress[
    from] + __post._numNFTPerAddress[to])
400 */
401 function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory
    data, bool safe) internal {
402     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
403     bool authorized = msg.sender == from ||
404         metaTx ||
405         _superOperators[msg.sender] ||
406         _operatorsForAll[from][msg.sender];
407
408     require(from != address(0), "from is zero address");
409     require(to != address(0), "can't send to zero address");
410
411     uint256 numTokens = ids.length;
412     /*@CTK "_batchTransferFrom_loop"
413     @pre from != address(0)
414     @pre to != address(0)
415     @pre numTokens < 5
416     @inv this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
417     @inv this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
418     @inv ids == ids__pre
419     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from
        )
420     @pre forall j: uint. (j >= 0 /\ j < numTokens) -> ((msg.sender == from) || (this.
        _metaTransactionContracts[msg.sender]) || (this._superOperators[msg.sender]) ||
        (this._operatorsForAll[from][msg.sender])) || (((this._owners[ids[j]] /
        2**255) == 1) && (this._operators[ids[j]] == msg.sender))
421     @inv i <= numTokens
422     @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
423     @inv numTokens == numTokens__pre
424     @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre.
        _numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
425     @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
426     @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
427     @post i == numTokens
428     @post !_should_return
429     */
430     for(uint256 i = 0; i < numTokens; i++) {
431         uint256 id = ids[i];
432         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
433         require(owner == from, "not owner in batchTransferFrom");
434         require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not
            authorized");
435         _owners[id] = uint256(to);
436         // emit Transfer(from, to, id);
437     }
438     if (from != to) {
439         _numNFTPerAddress[from] -= numTokens;
440         _numNFTPerAddress[to] += numTokens;
441     }
442     if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to,
        ERC721_MANDATORY_RECEIVER))) {
443         require(
444             _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
445             "erc721 batch transfer rejected by to"

```

```

446         );
447     }
448 }
449
450 /**
451  * @notice Transfer many tokens between 2 addresses ensuring the receiving contract has
452  *         a receiver method
453  * @param from The sender of the token
454  * @param to The recipient of the token
455  * @param ids The ids of the tokens
456  * @param data additional data
457  */
458 function safeBatchTransferFrom(address from, address to, uint256[] calldata ids, bytes
459     calldata data) external {
460     _batchTransferFrom(from, to, ids, data, true);
461 }
462
463 /**
464  * @notice Check if the contract supports an interface
465  * @param id The id of the interface
466  * @return True if the interface is supported
467  */
468 /*@CTK supportsInterface
469   @tag assume_completion
470   @post __return == (id == 0x01ffc9a7) || (id == 0x80ac58cd)
471  */
472 function supportsInterface(bytes4 id) external pure returns (bool) {
473     return id == 0x01ffc9a7 || id == 0x80ac58cd;
474 }
475
476 /**
477  * @notice Set the approval for an operator to manage all the tokens of the sender
478  * @param sender The address giving the approval
479  * @param operator The address receiving the approval
480  * @param approved The determination of the approval
481  */
482 //@CTK NO_OVERFLOW
483 //@CTK NO_BUF_OVERFLOW
484 //@CTK NO_ASF
485 /*@CTK _setApprovalForAll_require
486   @tag assume_completion
487   @post sender != address(0)
488   @post msg.sender == sender \/_metaTransactionContracts[msg.sender] == true \/_
489     _superOperators[msg.sender] == true
490   @post _superOperators[operator] == false
491  */
492 /*@CTK _setApprovalForAll_change
493   @tag assume_completion
494   @pre sender != address(0)
495   @pre msg.sender == sender \/_metaTransactionContracts[msg.sender] == true \/_
496     _superOperators[msg.sender] == true
497   @pre _superOperators[operator] == false
498   @post __post._operatorsForAll[sender][operator] == approved
499  */
500 function setApprovalForAllFor(
501     address sender,

```

```

500     address operator,
501     bool approved
502 ) external {
503     require(sender != address(0), "Invalid sender address");
504     require(
505         msg.sender == sender ||
506         _metaTransactionContracts[msg.sender] ||
507         _superOperators[msg.sender],
508         "not authorized to approve for all"
509     );
510
511     _setApprovalForAll(sender, operator, approved);
512 }
513
514 /**
515  * @notice Set the approval for an operator to manage all the tokens of the sender
516  * @param operator The address receiving the approval
517  * @param approved The determination of the approval
518  */
519 //@CTK NO_OVERFLOW
520 //@CTK NO_BUF_OVERFLOW
521 //@CTK NO_ASF
522 /*@CTK setApprovalForAll_require
523   @tag assume_completion
524   @post _superOperators[operator] == false
525 */
526 /*@CTK setApprovalForAll_change
527   @tag assume_completion
528   @pre _superOperators[operator] == false
529   @post __post._operatorsForAll[msg.sender][operator] == approved
530 */
531 function setApprovalForAll(address operator, bool approved) external {
532     _setApprovalForAll(msg.sender, operator, approved);
533 }
534
535 //@CTK NO_OVERFLOW
536 //@CTK NO_BUF_OVERFLOW
537 //@CTK NO_ASF
538 /*@CTK _setApprovalForAll_require
539   @tag assume_completion
540   @post _superOperators[operator] == false
541 */
542 /*@CTK _setApprovalForAll_change
543   @tag assume_completion
544   @pre _superOperators[operator] == false
545   @post __post._operatorsForAll[sender][operator] == approved
546 */
547 function _setApprovalForAll(
548     address sender,
549     address operator,
550     bool approved
551 ) internal {
552     require(
553         !_superOperators[operator],
554         "super operator can't have their approvalForAll changed"
555     );
556     _operatorsForAll[sender][operator] = approved;
557 }

```

```

558     emit ApprovalForAll(sender, operator, approved);
559 }
560
561 /**
562  * @notice Check if the sender approved the operator
563  * @param owner The address of the owner
564  * @param operator The address of the operator
565  * @return The status of the approval
566  */
567 //CTK NO_OVERFLOW
568 //CTK NO_BUF_OVERFLOW
569 //CTK NO_ASF
570 /*CTK isApprovedForAll
571     @post (_operatorsForAll[owner][operator] == true /\ _superOperators[operator] == true)
572         -> __return == true
573     @post (_operatorsForAll[owner][operator] == false /\ _superOperators[operator] ==
574         false) -> __return == false
575 */
576 function isApprovedForAll(address owner, address operator)
577     external
578     view
579     returns (bool)
580 {
581     return _operatorsForAll[owner][operator] || _superOperators[operator];
582 }
583
584 //CTK FAIL NO_OVERFLOW
585 //CTK NO_BUF_OVERFLOW
586 //CTK NO_ASF
587 /*CTK _burn_require
588     @tag assume_completion
589     @post from == owner
590 */
591 /*CTK _burn_change
592     @tag assume_completion
593     @pre from == owner
594     @post __post._owners[id] == 2**160
595     @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
596 */
597 function _burn(address from, address owner, uint256 id) public {
598     require(from == owner, "not owner");
599     _owners[id] = 2**160; // cannot mint it again
600     _numNFTPerAddress[from]--;
601     emit Transfer(from, address(0), id);
602 }
603
604 /// @notice Burns token `id`.
605 /// @param id token which will be burnt.
606 //CTK FAIL NO_OVERFLOW
607 //CTK NO_BUF_OVERFLOW
608 //CTK NO_ASF
609 /*CTK burn_require
610     @tag assume_completion
611     @post msg.sender == address(_owners[id])
612 */
613 /*CTK burn_change
614     @tag assume_completion
615     @pre msg.sender == address(_owners[id])

```

```

614     @post __post._owners[id] == 2**160
615     @post __post._numNFTPerAddress[msg.sender] == _numNFTPerAddress[msg.sender] - 1
616     */
617     function burn(uint256 id) external {
618         _burn(msg.sender, _ownerOf(id), id);
619     }
620
621     /// @notice Burn token`id` from `from`.
622     /// @param from address whose token is to be burnt.
623     /// @param id token which will be burnt.
624     ///@CTK NO_OVERFLOW
625     ///@CTK NO_BUF_OVERFLOW
626     /*@CTK burnFrom_require
627         @tag assume_completion
628         @post from != address(0)
629         @post (msg.sender == from) || _metaTransactionContracts[msg.sender] || ((_owners[id] /
630             2**255) == 1 && _operators[id] == msg.sender) || _superOperators[msg.sender] ||
631             _operatorsForAll[from][msg.sender]
632         @post from == address(_owners[id])
633     */
634     /*@CTK burnFrom_change
635         @tag assume_completion
636         @pre from != address(0)
637         @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || ((_owners[id] /
638             2**255) == 1 && _operators[id] == msg.sender) || _superOperators[msg.sender] ||
639             _operatorsForAll[from][msg.sender]
640         @pre from == address(_owners[id])
641         @post __post._owners[id] == 2**160
642         @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
643     */
644     function burnFrom(address from, uint256 id) external {
645         require(from != address(0), "Invalid sender address");
646         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
647         require(
648             msg.sender == from ||
649             _metaTransactionContracts[msg.sender] ||
650             (operatorEnabled && _operators[id] == msg.sender) ||
651             _superOperators[msg.sender] ||
652             _operatorsForAll[from][msg.sender],
653             "not authorized to burn"
654         );
655         _burn(from, owner, id);
656     }
657
658     function _checkOnERC721Received(address operator, address from, address to, uint256
659         tokenId, bytes memory _data)
660         internal returns (bool)
661     {
662         bytes4 retval = ERC721TokenReceiver(to).onERC721Received(operator, from, tokenId,
663             _data);
664         return (retval == _ERC721_RECEIVED);
665     }
666
667     function _checkOnERC721BatchReceived(address operator, address from, address to, uint256
668         [] memory ids, bytes memory _data)
669         internal returns (bool)
670     {
671         bytes4 retval = ERC721MandatoryTokenReceiver(to).onERC721BatchReceived(operator,

```



```

        from, ids, _data);
665     return (retval == _ERC721_BATCH_RECEIVED);
666 }
667 }

```

File SuperOperators.sol

```

1  pragma solidity ^0.5.2;
2
3  import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/Admin.sol";
4
5  contract SuperOperators is Admin {
6
7      mapping(address => bool) internal _superOperators;
8
9      event SuperOperator(address superOperator, bool enabled);
10
11     /// @notice Enable or disable the ability of `superOperator` to transfer tokens of all (
12         superOperator rights).
13     /// @param superOperator address that will be given/removed superOperator right.
14     /// @param enabled set whether the superOperator is enabled or disabled.
15     ///@CTK NO_OVERFLOW
16     ///@CTK NO_BUF_OVERFLOW
17     ///@CTK NO_ASF
18     /*@CTK setSuperOperator_admin
19         @tag assume_completion
20         @inv msg.sender == _admin
21     */
22     /*@CTK setSuperOperator_change
23         @tag assume_completion
24         @pre msg.sender == _admin
25         @post __post._superOperators[superOperator] == enabled
26     */
27     function setSuperOperator(address superOperator, bool enabled) external {
28         require(
29             msg.sender == _admin,
30             "only admin is allowed to add super operators"
31         );
32         _superOperators[superOperator] = enabled;
33         emit SuperOperator(superOperator, enabled);
34     }
35
36     /// @notice check whether address `who` is given superOperator rights.
37     /// @param who The address to query.
38     /// @return whether the address has superOperator rights.
39     ///@CTK NO_OVERFLOW
40     ///@CTK NO_BUF_OVERFLOW
41     ///@CTK NO_ASF
42     /*@CTK isSuperOperator
43         @tag assume_completion
44         @post __return == _superOperators[who]
45     */
46     function isSuperOperator(address who) public view returns (bool) {
47         return _superOperators[who];
48     }
49 }

```

File AddressUtils.sol

```

1  pragma solidity ^0.5.2;

```

```

2
3 library AddressUtils {
4
5     //@CTK NO_OVERFLOW
6     //@CTK NO_BUF_OVERFLOW
7     //@CTK NO_ASF
8     function toPayable(address _address) internal pure returns (address payable _payable) {
9         return address(uint160(_address));
10    }
11
12    function isContract(address addr) internal view returns (bool) {
13        // for accounts without code, i.e. `keccak256('')`:
14        bytes32 accountHash = 0
15            xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
16
17        bytes32 codehash;
18        // solium-disable-next-line security/no-inline-assembly
19        assembly {
20            codehash := extcodehash(addr)
21        }
22        return (codehash != 0x0 && codehash != accountHash);
23    }
24 }

```

File Land.sol

```

1 /* solhint-disable no-empty-blocks */
2
3 pragma solidity 0.5.9;
4
5 import "../sandbox-private-contracts/src/Land/erc721/LandBaseToken.sol";
6
7 contract Land is LandBaseToken {
8     //@CTK NO_OVERFLOW
9     //@CTK NO_BUF_OVERFLOW
10    //@CTK NO_ASF
11    /*@CTK Land
12        @tag assume_completion
13        @post __post._admin == admin
14        @post __post._metaTransactionContracts[metaTransactionContract] == true
15    */
16    constructor(
17        address metaTransactionContract,
18        address admin
19    ) public LandBaseToken(
20        metaTransactionContract,
21        admin
22    ) {
23    }
24
25    /**
26     * @notice Return the name of the token contract
27     * @return The name of the token contract
28     */
29    /*@CTK name
30        @post __return == "Sandbox's LANDs"
31    */
32    function name() external pure returns (string memory) {
33        return "Sandbox's LANDs";
34    }
35 }

```

```

34     }
35
36     /**
37     * @notice Return the symbol of the token contract
38     * @return The symbol of the token contract
39     */
40     /*@CTK symbol
41     @post __return == "LAND"
42     */
43     function symbol() external pure returns (string memory) {
44         return "LAND";
45     }
46
47     // solium-disable-next-line security/no-assign-params
48     //TODO
49     function uint2str(uint _i) internal pure returns (string memory) {
50         if (_i == 0) {
51             return "0";
52         }
53         uint j = _i;
54         uint len;
55         while (j != 0) {
56             len++;
57             j /= 10;
58         }
59         bytes memory bstr = new bytes(len);
60         uint k = len - 1;
61         while (_i != 0) {
62             bstr[k--] = byte(uint8(48 + _i % 10));
63             _i /= 10;
64         }
65         return string(bstr);
66     }
67
68     /**
69     * @notice Return the URI of a specific token
70     * @param id The id of the token
71     * @return The URI of the token
72     */
73     function tokenURI(uint256 id) public view returns (string memory) {
74         require(_ownerOf(id) != address(0), "Id does not exist");
75         return
76             string(
77                 abi.encodePacked(
78                     "https://api.sandbox.game/lands/",
79                     uint2str(id),
80                     "/metadata.json"
81                 )
82             );
83     }
84
85     /**
86     * @notice Check if the contract supports an interface
87     * 0x01ffc9a7 is ERC-165
88     * 0x80ac58cd is ERC-721
89     * 0x5b5e139f is ERC-721 metadata
90     * @param id The id of the interface
91     * @return True if the interface is supported

```

```

92     */
93     //@CTK NO_OVERFLOW
94     //@CTK NO_BUF_OVERFLOW
95     //@CTK NO_ASF
96     /*@CTK supportsInterface
97         @tag assume_completion
98         @post (id == 0x01ffc9a7 /\ id == 0x80ac58cd /\ id == 0x5b5e139f) -> __return == true
99         @post (id != 0x01ffc9a7 /\ id != 0x80ac58cd /\ id != 0x5b5e139f) -> __return == false
100     */
101     function supportsInterface(bytes4 id) external pure returns (bool) {
102         return id == 0x01ffc9a7 || id == 0x80ac58cd || id == 0x5b5e139f;
103     }
104 }

```

File LandBaseToken.sol

```

1 /* solhint-disable func-order, code-complexity */
2 pragma solidity 0.5.9;
3
4 import "./ERC721BaseToken.sol";
5
6 contract LandBaseToken is ERC721BaseToken {
7     // Our grid is 408 x 408 lands
8     uint256 internal constant GRID_SIZE = 408;
9
10    uint256 internal constant LAYER =      0
        xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF;
11    uint256 internal constant LAYER_1x1 = 0
        x0000000000000000000000000000000000000000000000000000000000000000;
12    uint256 internal constant LAYER_3x3 = 0
        x0100000000000000000000000000000000000000000000000000000000000000;
13    uint256 internal constant LAYER_6x6 = 0
        x0200000000000000000000000000000000000000000000000000000000000000;
14    uint256 internal constant LAYER_12x12 = 0
        x0300000000000000000000000000000000000000000000000000000000000000;
15    uint256 internal constant LAYER_24x24 = 0
        x0400000000000000000000000000000000000000000000000000000000000000;
16
17    mapping(address => bool) internal _minters;
18    event Minter(address superOperator, bool enabled);
19
20    /// @notice Enable or disable the ability of `minter` to mint tokens
21    /// @param minter address that will be given/removed minter right.
22    /// @param enabled set whether the minter is enabled or disabled.
23    //@CTK NO_OVERFLOW
24    //@CTK NO_BUF_OVERFLOW
25    //@CTK NO_ASF
26    /*@CTK setMinter_require
        @tag assume_completion
        @post msg.sender == _admin
        */
27    /*@CTK setMinter_change
        @tag assume_completion
        @post __post._minters[minter] == enabled
        */
28
29    function setMinter(address minter, bool enabled) external {
30        require(
31            msg.sender == _admin,
32            "only admin is allowed to add minters"
33        );
34    }

```

```

38     );
39     _minters[minter] = enabled;
40     emit Minter(minter, enabled);
41 }
42
43 /// @notice check whether address `who` is given minter rights.
44 /// @param who The address to query.
45 /// @return whether the address has minter rights.
46 ///@CTK NO_OVERFLOW
47 ///@CTK NO_BUF_OVERFLOW
48 ///@CTK NO_ASF
49 /*@CTK isMinter
50     @tag assume_completion
51     @post __return == _minters[who]
52 */
53 function isMinter(address who) public view returns (bool) {
54     return _minters[who];
55 }
56
57 ///@CTK NO_OVERFLOW
58 ///@CTK NO_BUF_OVERFLOW
59 ///@CTK NO_ASF
60 /*@CTK LandBaseToken
61     @tag assume_completion
62     @post __post._admin == admin
63     @post __post._metaTransactionContracts[metaTransactionContract] == true
64 */
65 constructor(
66     address metaTransactionContract,
67     address admin
68 ) public ERC721BaseToken(metaTransactionContract, admin) {
69 }
70
71 /// @notice total width of the map
72 /// @return width
73 /*@CTK width
74     @post __return == GRID_SIZE
75 */
76 function width() external returns(uint256) {
77     return GRID_SIZE;
78 }
79
80 /// @notice total height of the map
81 /// @return height
82 /*@CTK height
83     @post __return == GRID_SIZE
84 */
85 function height() external returns(uint256) {
86     return GRID_SIZE;
87 }
88
89 /// @notice x coordinate of Land token
90 /// @param id tokenId
91 /// @return the x coordinates
92 ///@CTK NO_OVERFLOW
93 ///@CTK NO_BUF_OVERFLOW
94 ///@CTK NO_ASF
95 /*@CTK x

```

```

96     @tag assume_completion
97     @pre GRID_SIZE == 408
98     @pre address(_owners[id]) != address(0)
99     @post __return == id % GRID_SIZE
100    */
101    function x(uint256 id) external returns(uint256) {
102        require(_ownerOf(id) != address(0), "token does not exist");
103        return id % GRID_SIZE;
104    }
105
106    /// @notice y coordinate of Land token
107    /// @param id tokenId
108    /// @return the y coordinates
109    ///@CTK NO_OVERFLOW
110    ///@CTK NO_BUF_OVERFLOW
111    ///@CTK NO_ASF
112    /*CTK y
113    @tag assume_completion
114    @pre GRID_SIZE == 408
115    @pre address(_owners[id]) != address(0)
116    @post __return == id / GRID_SIZE
117    */
118    function y(uint256 id) external returns(uint256) {
119        require(_ownerOf(id) != address(0), "token does not exist");
120        return id / GRID_SIZE;
121    }
122
123    /**
124     * @notice Mint a new quad (aligned to a quad tree with size 3, 6, 12 or 24 only)
125     * @param to The recipient of the new quad
126     * @param size The size of the new quad
127     * @param x The top left x coordinate of the new quad
128     * @param y The top left y coordinate of the new quad
129     * @param data extra data to pass to the transfer
130     */
131    ///@CTK FAIL NO_OVERFLOW
132    ///@CTK NO_BUF_OVERFLOW
133    ///@CTK NO_ASF
134    /*CTK mintQuad_require
135    @tag assume_completion
136    @pre GRID_SIZE == 408
137    @post to != address(0)
138    @post _minters[msg.sender] == true
139    @post (x % size == 0) /\ (y % size == 0)
140    @post (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
141    @post (size == 1 /\ size == 3 /\ size == 6 /\ size == 12 /\ size == 24)
142    @post _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0
143    @post size <= 12 -> _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] ==
        0
144    @post size <= 6 -> _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0
145    @post size <= 3 -> _owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0
146    */
147    /*CTK mintQuad_change
148    @tag assume_completion
149    @pre GRID_SIZE == 408
150    @pre LAYER == 0xFF00000000000000000000000000000000000000000000000000000000000000
151    @pre LAYER_1x1 == 0x00000000000000000000000000000000000000000000000000000000000000
152    @pre LAYER_3x3 == 0x01000000000000000000000000000000000000000000000000000000000000

```

page 300

```

209         for (uint256 y12i = y; y12i < toY; y12i += 12) {
210             uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
211             require(_owners[id12x12] == 0, "Already minted as 12x12");
212         }
213     }
214 }
215
216 if (size <= 6) {
217     require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
        minted as 6x6");
218 } else {
219     for (uint256 x6i = x; x6i < toX; x6i += 6) {
220         for (uint256 y6i = y; y6i < toY; y6i += 6) {
221             uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
222             require(_owners[id6x6] == 0, "Already minted as 6x6");
223         }
224     }
225 }
226
227 if (size <= 3) {
228     require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
        minted as 3x3");
229 } else {
230     for (uint256 x3i = x; x3i < toX; x3i += 3) {
231         for (uint256 y3i = y; y3i < toY; y3i += 3) {
232             uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
233             require(_owners[id3x3] == 0, "Already minted as 3x3");
234         }
235     }
236 }
237
238 /*@CTK mintQuad_loopx
239    @tag assume_completion
240    @pre GRID_SIZE == 408
241    @inv i <= size * size
242    @post i == size * size
243    */
244 for (uint256 i = 0; i < size*size; i++) {
245     uint256 id = _idInPath(i, size, x, y);
246     require(_owners[id] == 0, "Already minted");
247     emit Transfer(address(0), to, id);
248 }
249
250 _owners[quadId] = uint256(to);
251 _numNFTPerAddress[to] += size * size;
252
253 _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
254 }
255
256 //@CTK FAIL NO_OVERFLOW
257 //@CTK NO_BUF_OVERFLOW
258 //@CTK FAIL NO_ASF
259 /*@CTK _idInPath
260    @tag assume_completion
261    @pre GRID_SIZE == 408
262    @post (((i / size) % 2) == 0) -> __return == (x + (i%size)) + ((y + i / size) *
        GRID_SIZE)
263    @post (((i / size) % 2) == 1) -> __return == ((x + size) - (1 + i%size)) + ((y + i /

```



```

        size) * GRID_SIZE)
264 */
265 function _idInPath(uint256 i, uint256 size, uint256 x, uint256 y) internal pure returns(
    uint256) {
266     uint256 row = i / size;
267     if(row % 2 == 0) { // allow ids to follow a path in a quad
268         return (x + (i%size)) + ((y + row) * GRID_SIZE);
269     } else {
270         return ((x + size) - (1 + i%size)) + ((y + row) * GRID_SIZE);
271     }
272 }
273
274 /// @notice transfer one quad (aligned to a quad tree with size 3, 6, 12 or 24 only)
275 /// @param from current owner of the quad
276 /// @param to destination
277 /// @param size size of the quad
278 /// @param x The top left x coordinate of the quad
279 /// @param y The top left y coordinate of the quad
280 /// @param data additional data
281 ///@CTK FAIL NO_OVERFLOW
282 ///@CTK NO_BUF_OVERFLOW
283 ///@CTK NO_ASF
284 /*@CTK transferQuad_require
285     @tag assume_completion
286     @post from != address(0)
287     @post to != address(0)
288     @post (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
        _superOperators[msg.sender] /\ _operatorsForAll[from][msg.sender])
289 */
290 /*@CTK transferQuad_change
291     @tag assume_completion
292     @pre from != to
293     @pre from != address(0)
294     @pre to != address(0)
295     @pre (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
        _superOperators[msg.sender] /\ _operatorsForAll[from][msg.sender])
296     @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - size * size
297     @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + size * size
298 */
299 function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
    bytes calldata data) external {
300     require(from != address(0), "from is zero address");
301     require(to != address(0), "can't send to zero address");
302     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
303     if (msg.sender != from && !metaTx) {
304         require(
305             _superOperators[msg.sender] ||
306             _operatorsForAll[from][msg.sender],
307             "not authorized to transferQuad"
308         );
309     }
310     _transferQuad(from, to, size, x, y);
311     _numNFTPerAddress[from] -= size * size;
312     _numNFTPerAddress[to] += size * size;
313
314     _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
        );
315 }

```

```

316
317 /*@CTK _checkBatchReceiverAcceptQuad
318     @tag assume_completion
319     @pre size >= 1
320     @pre GRID_SIZE == 408
321 */
322 function _checkBatchReceiverAcceptQuad(
323     address operator,
324     address from,
325     address to,
326     uint256 size,
327     uint256 x,
328     uint256 y,
329     bytes memory data
330 ) internal {
331     if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
332         uint256[] memory ids = new uint256[](size*size);
333         /*@CTK _checkBatchReceiverAcceptQuad_forloop
334             @inv i <= size * size
335             @pre size >= 1
336             @pre GRID_SIZE == 408
337             @post i == size * size
338             @post !__should_return
339         */
340         for (uint256 i = 0; i < size*size; i++) {
341             ids[i] = _idInPath(i, size, x, y);
342         }
343         require(
344             _checkOnERC721BatchReceived(operator, from, to, ids, data),
345             "erc721 batch transfer rejected by to"
346         );
347     }
348 }
349
350 /// @notice transfer multiple quad (aligned to a quad tree with size 3, 6, 12 or 24 only
351     )
352     /// @param from current owner of the quad
353     /// @param to destination
354     /// @param sizes list of sizes for each quad
355     /// @param xs list of top left x coordinates for each quad
356     /// @param ys list of top left y coordinates for each quad
357     /// @param data additional data
358     ///@CTK NO_OVERFLOW
359     ///@CTK NO_BUF_OVERFLOW
360     ///@CTK NO_ASF
361     /*@CTK transferQuad_require
362         @tag assume_completion
363         @post from != address(0)
364         @post to != address(0)
365         @post (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
366             _superOperators[msg.sender] /\ _operatorsForAll[from][msg.sender])
367     */
368 function batchTransferQuad(
369     address from,
370     address to,
371     uint256[] calldata sizes,
372     uint256[] calldata xs,
373     uint256[] calldata ys,

```

```

372     bytes calldata data
373 ) external {
374     require(from != address(0), "from is zero address");
375     require(to != address(0), "can't send to zero address");
376     require(sizes.length == xs.length && xs.length == ys.length, "invalid data");
377     bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
378     if (msg.sender != from && !metaTx) {
379         require(
380             _superOperators[msg.sender] ||
381             _operatorsForAll[from][msg.sender],
382             "not authorized to transferMultiQuads"
383         );
384     }
385     uint256 numTokensTransferred = 0;
386     for (uint256 i = 0; i < sizes.length; i++) {
387         uint256 size = sizes[i];
388         _transferQuad(from, to, size, xs[i], ys[i]);
389         numTokensTransferred += size * size;
390     }
391     _numNFTPerAddress[from] -= numTokensTransferred;
392     _numNFTPerAddress[to] += numTokensTransferred;
393
394     if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
395         uint256[] memory ids = new uint256[](numTokensTransferred);
396         uint256 counter = 0;
397         for (uint256 j = 0; j < sizes.length; j++) {
398             uint256 size = sizes[j];
399             for (uint256 i = 0; i < size*size; i++) {
400                 ids[counter] = _idInPath(i, size, xs[j], ys[j]);
401                 counter++;
402             }
403         }
404         require(
405             _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
406             "erc721 batch transfer rejected by to"
407         );
408     }
409 }
410
411 // @CTK NO_BUF_OVERFLOW
412 // @CTK NO_ASF
413 /* @CTK _transferQuad_require
414     @tag assume_completion
415     @pre GRID_SIZE == 408
416     @post (size == 1) -> (address(_owners[x + y * GRID_SIZE]) != address(0) /\ address(
417         _owners[x + y * GRID_SIZE]) == from)
418 */
419 /* @CTK _transferQuad_change
420     @tag assume_completion
421     @pre GRID_SIZE == 408
422     @pre (size == 1) -> (address(_owners[x + y * GRID_SIZE]) != address(0) /\ address(
423         _owners[x + y * GRID_SIZE]) == from)
424     @post (size == 1) -> (_owners[x + y * GRID_SIZE] == uint256(to))
425 */
426 function _transferQuad(address from, address to, uint256 size, uint256 x, uint256 y)
    internal {
    if (size == 1) {
        uint256 id1x1 = x + y * GRID_SIZE;
    }
}

```

```

427     address owner = _ownerOf(id1x1);
428     require(owner != address(0), "token does not exist");
429     require(owner == from, "not owner in _transferQuad");
430     _owners[id1x1] = uint256(to);
431 } else {
432     _regroup(from, to, size, x, y);
433 }
434 /*@CTK _transferQuad_loop
435   @inv i <= size * size
436   @post i == size * size
437   */
438 for (uint256 i = 0; i < size*size; i++) {
439     emit Transfer(from, to, _idInPath(i, size, x, y));
440 }
441 }
442
443 //@CTK NO_OVERFLOW
444 //@CTK NO_BUF_OVERFLOW
445 //@CTK NO_ASF
446 /*@CTK _checkAndClear_require
447   @tag assume_completion
448   @post (_owners[id] != 0) -> (address(_owners[id]) == from)
449   */
450 /*@CTK _checkAndClear_change
451   @tag assume_completion
452   @pre (_owners[id] != 0) -> (address(_owners[id]) == from)
453   @post _owners[id] == 0 -> __return == false
454   @post (_owners[id] != 0) -> __post._owners[id] == 0
455   @post (_owners[id] != 0) -> __return == true
456   */
457 function _checkAndClear(address from, uint256 id) internal returns(bool) {
458     uint256 owner = _owners[id];
459     if (owner != 0) {
460         require(address(owner) == from, "not owner");
461         _owners[id] = 0;
462         return true;
463     }
464     return false;
465 }
466
467 //@CTK FAIL NO_OVERFLOW
468 //@CTK NO_BUF_OVERFLOW
469 //@CTK NO_ASF
470 /*@CTK _regroup_require
471   @tag assume_completion
472   @post (x % size == 0) /\ (y % size == 0)
473   @post (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
474   @post (size == 1 /\ size == 3 /\ size == 6 /\ size == 12 /\ size == 24)
475   */
476 function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
477 {
478     require(x % size == 0 && y % size == 0, "Invalid coordinates");
479     require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
480     if (size == 3) {
481         _regroup3x3(from, to, x, y, true);
482     } else if (size == 6) {
483         _regroup6x6(from, to, x, y, true);

```

```

484     } else if (size == 12) {
485         _regroup12x12(from, to, x, y, true);
486     } else if (size == 24) {
487         _regroup24x24(from, to, x, y, true);
488     } else {
489         require(false, "Invalid size");
490     }
491 }
492
493 //@CTK FAIL NO_OVERFLOW
494 //@CTK NO_BUF_OVERFLOW
495 //@CTK NO_ASF
496 /*@CTK _regroup3x3_require
497   @tag assume_completion
498   @pre GRID_SIZE == 408
499   @post (set == true /\ ownerOfAll == false) -> (_owners[LAYER_3x3 + x + y * GRID_SIZE]
    == uint256(from) /\ _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] ==
    uint256(from) /\ _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] ==
    uint256(from) /\ _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE]
    == uint256(from))
500 */
501 function _regroup3x3(address from, address to, uint256 x, uint256 y, bool set) internal
    returns (bool) {
502     uint256 id = x + y * GRID_SIZE;
503     uint256 quadId = LAYER_3x3 + id;
504     bool ownerOfAll = true;
505     for (uint256 xi = x; xi < x+3; xi++) {
506         for (uint256 yi = y; yi < y+3; yi++) {
507             ownerOfAll = _checkAndClear(from, xi + yi * GRID_SIZE) && ownerOfAll;
508         }
509     }
510     if(set) {
511         if(!ownerOfAll) {
512             require(
513                 _owners[quadId] == uint256(from) ||
514                 _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == uint256(from)
                    ||
515                 _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
                    from) ||
516                 _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
                    from),
                    "not owner of all sub quads nor parent quads"
                    );
517         }
518     }
519     _owners[quadId] = uint256(to);
520     return true;
521 }
522
523 return ownerOfAll;
524 }
525
526 //@CTK FAIL NO_OVERFLOW
527 //@CTK NO_BUF_OVERFLOW
528 //@CTK NO_ASF
529 function _regroup6x6(address from, address to, uint256 x, uint256 y, bool set) internal
    returns (bool) {
530     uint256 id = x + y * GRID_SIZE;
531     uint256 quadId = LAYER_6x6 + id;
532     bool ownerOfAll = true;

```

```

533     for (uint256 xi = x; xi < x+6; xi += 3) {
534         for (uint256 yi = y; yi < y+6; yi += 3) {
535             bool ownAllIndividual = _regroup3x3(from, to, xi, yi, false);
536             uint256 id3x3 = LAYER_3x3 + xi + yi * GRID_SIZE;
537             uint256 owner3x3 = _owners[id3x3];
538             if (owner3x3 != 0) {
539                 if(!ownAllIndividual) {
540                     require(owner3x3 == uint256(from), "not owner of 3x3 quad");
541                 }
542                 _owners[id3x3] = 0;
543             }
544             ownerOfAll = (ownAllIndividual || owner3x3 != 0) && ownerOfAll;
545         }
546     }
547     if(set) {
548         if(!ownerOfAll) {
549             require(
550                 _owners[quadId] == uint256(from) ||
551                 _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == uint256(
552                     from) ||
553                 _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
554                     from),
555                 "not owner of all sub quads nor parent quads"
556             );
557             _owners[quadId] = uint256(to);
558             return true;
559         }
560     }
561     return ownerOfAll;
562 }
563
564 // @CTK FAIL NO_OVERFLOW
565 // @CTK NO_BUF_OVERFLOW
566 // @CTK NO_ASF
567
568 function _regroup12x12(address from, address to, uint256 x, uint256 y, bool set)
569     internal returns (bool) {
570     uint256 id = x + y * GRID_SIZE;
571     uint256 quadId = LAYER_12x12 + id;
572     bool ownerOfAll = true;
573     for (uint256 xi = x; xi < x+12; xi += 6) {
574         for (uint256 yi = y; yi < y+12; yi += 6) {
575             bool ownAllIndividual = _regroup6x6(from, to, xi, yi, false);
576             uint256 id6x6 = LAYER_6x6 + xi + yi * GRID_SIZE;
577             uint256 owner6x6 = _owners[id6x6];
578             if (owner6x6 != 0) {
579                 if(!ownAllIndividual) {
580                     require(owner6x6 == uint256(from), "not owner of 6x6 quad");
581                 }
582                 _owners[id6x6] = 0;
583             }
584             ownerOfAll = (ownAllIndividual || owner6x6 != 0) && ownerOfAll;
585         }
586     }
587     if(set) {
588         if(!ownerOfAll) {
589             require(
590                 _owners[quadId] == uint256(from) ||
591                 _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(

```

```

        from),
588         "not owner of all sub quads nor parent quads"
589     );
590 }
591 _owners[quadId] = uint256(to);
592 return true;
593 }
594 return ownerOfAll;
595 }
596
597 //CTK FAIL NO_OVERFLOW
598 //CTK NO_BUF_OVERFLOW
599 //CTK NO_ASF
600 function _regroup24x24(address from, address to, uint256 x, uint256 y, bool set)
    internal returns (bool) {
601     uint256 id = x + y * GRID_SIZE;
602     uint256 quadId = LAYER_24x24 + id;
603     bool ownerOfAll = true;
604     for (uint256 xi = x; xi < x+24; xi += 12) {
605         for (uint256 yi = y; yi < y+24; yi += 12) {
606             bool ownAllIndividual = _regroup12x12(from, to, xi, yi, false);
607             uint256 id12x12 = LAYER_12x12 + xi + yi * GRID_SIZE;
608             uint256 owner12x12 = _owners[id12x12];
609             if (owner12x12 != 0) {
610                 if(!ownAllIndividual) {
611                     require(owner12x12 == uint256(from), "not owner of 12x12 quad");
612                 }
613                 _owners[id12x12] = 0;
614             }
615             ownerOfAll = (ownAllIndividual || owner12x12 != 0) && ownerOfAll;
616         }
617     }
618     if(set) {
619         if(!ownerOfAll) {
620             require(
621                 _owners[quadId] == uint256(from),
622                 "not owner of all sub quads not parent quad"
623             );
624         }
625         _owners[quadId] = uint256(to);
626         return true;
627     }
628     return ownerOfAll || _owners[quadId] == uint256(from);
629 }
630
631 //CTK NO_OVERFLOW
632 //CTK NO_BUF_OVERFLOW
633 //CTK NO_ASF
634 /*CTK FAIL "_ownerOf"
635     @pre GRID_SIZE == 408
636     @pre (id & LAYER) == 0
637     @post (_owners[id] != 0) -> (__return == address(_owners[id]))
638 */
639 function _ownerOf(uint256 id) internal view returns (address) {
640     require(id & LAYER == 0, "Invalid token id");
641     uint256 x = id % GRID_SIZE;
642     uint256 y = id / GRID_SIZE;
643     uint256 owner1x1 = _owners[id];

```

```

644
645     if (owner1x1 != 0) {
646         return address(owner1x1); // cast to zero
647     } else {
648         address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) *
        GRID_SIZE]);
649         if (owner3x3 != address(0)) {
650             return owner3x3;
651         } else {
652             address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) *
        GRID_SIZE]);
653             if (owner6x6 != address(0)) {
654                 return owner6x6;
655             } else {
656                 address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) *
        12) * GRID_SIZE]);
657                 if (owner12x12 != address(0)) {
658                     return owner12x12;
659                 } else {
660                     return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) *
        GRID_SIZE]);
661                 }
662             }
663         }
664     }
665 }
666
667 //@*CTK FAIL NO_OVERFLOW
668 //@*CTK NO_BUF_OVERFLOW
669 //@*CTK FAIL NO_ASF
670 /*@*CTK _ownerAndOperatorEnabledOf
671   @pre GRID_SIZE == 408
672   @pre (id & LAYER) == 0
673   @post owner == address(_owners[id])
674   @post operatorEnabled == ((_owners[id] / 2**255) == 1)
675 */
676 function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner,
    bool operatorEnabled) {
677     require(id & LAYER == 0, "Invalid token id");
678     uint256 x = id % GRID_SIZE;
679     uint256 y = id / GRID_SIZE;
680     uint256 owner1x1 = _owners[id];
681
682     if (owner1x1 != 0) {
683         owner = address(owner1x1);
684         operatorEnabled = (owner1x1 / 2**255) == 1;
685     } else {
686         address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) *
        GRID_SIZE]);
687         if (owner3x3 != address(0)) {
688             owner = owner3x3;
689             operatorEnabled = false;
690         } else {
691             address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) *
        GRID_SIZE]);
692             if (owner6x6 != address(0)) {
693                 owner = owner6x6;
694                 operatorEnabled = false;

```



```

695         } else {
696             address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) *
697                 12) * GRID_SIZE]);
698             if (owner12x12 != address(0)) {
699                 owner = owner12x12;
700                 operatorEnabled = false;
701             } else {
702                 owner = address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) *
703                     GRID_SIZE]);
704                 operatorEnabled = false;
705             }
706         }
707     }
708 }
709 }

```

