CERTIK

# 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

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 LandSale 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**

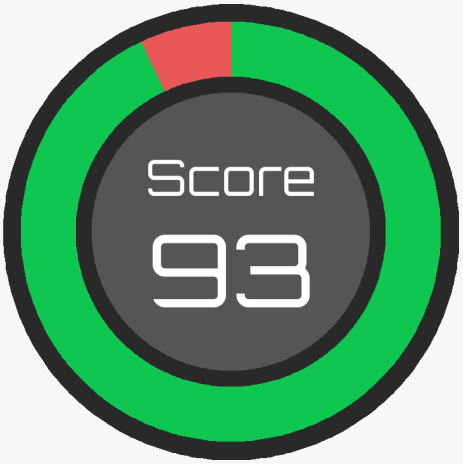
Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.

# 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 | An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type. | 0 | SWC-101 |
| Function Incorrectness | Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities. | 0 | |
| 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<br>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 Checksum[1]

- **AddressUtils.sol**
  `2a717cd56c8a3f562015bacb0ab7b6d93cb639d64221728520bf3f40217c8957`
- **Admin.sol**
  `f336e6bd77e29368a3afe4ffecdc9eafe0b2854f2c303d47405a45a85bfcfb6e`
- **ERC721BaseToken.sol**
  `aab7dd819e3606949889fb0511ebac0c6b9108ffc28503813a3f1dac3a26d230`
- **Land.sol**
  `049b1ad829349d3deeea557bb19a6e36708520b359551272b5fdd6869b34ec8d`
- **LandBaseToken.sol**
  `ebb6ab14f7766bc12a1d7c98566160d2ac4350c84c2294ba1d0f5623bcbbca48`
- **LandSale.sol**
  `cda06e36c1d2b17d98c114d9ba0425f043efe1514fa1a529dd89f3b43626dec6`
- **MetaTransactionReceiver.sol**
  `8bae54108e69e81fcffe22425c311814d7339e078ae37e9c1c67c30cf4e4a6e9`
- **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 97013*dcbcb*29032*ea*8*adba*065*a*9*e*490138*d*25713, previous

[1]Commit: 752*e*899*abe*7*d*5492227*d*28470*a*0*bc*2*a*0*ae*6*dfd*41

- 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 *eb2866621f04691b8037b1fca03d335ab29bb8e3*, 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. INFO `balanceOf()`: Function return style is not unified. `_balance` variable should be assigned.
   - (The Sandbox - updated): Fixed in commit *478d4b8391e9aba2f7e13fb66b6abeaaa7b22473*.
4. INFO `approveFor()`: The function should be extracted as a common internal function `_approveFor()`. Both public `approveFor()` and `approve()` should be using the same internal function to avoid multiple copy of the same code.
   - (The Sandbox - updated): Fixed in commit *478d4b8391e9aba2f7e13fb66b6abeaaa7b22473*.
5. 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.
6. 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.
7. INFO `selfTransferCounters`: The state variable should be removed as it is not being used.
   - (The Sandbox - updated): Fixed in commit *478d4b8391e9aba2f7e13fb66b6abeaaa7b22473*.

8. 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 outweight the need to ensure it is reset properly.

**Land.sol** commit *eb2866621f04691b8037b1fca03d335ab29bb8e3*, 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 *eb2866621f04691b8037b1fca03d335ab29bb8e3*, 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 Recommend using `uint256` instead of `uint16` for variables `x`, `y` and `size` considering gas saving.
   - (The Sandbox - updated): We changed the `uint16` to `uint256` as you recommended in commit *f7fad443b9a4730ead473598dbc7e36180871336*.
4. INFO `width()` and `height()`: Recommend marking these functions with `pure`.
5. INFO `x()` and `y()`: Recommend marking these functions with `view`.
6. MINOR `x()`, `y()`: Recommend using `SafeMath` for all arithmetic operations.
   - (The Sandbox - updated): We check for existence in commit *478d4b8391e9aba2f7e13fb66b6abeaaa7b22473*, which should not require any `SafeMath`.
7. MINOR `mintQuad()`: Recommend checking the validity of the recipient `to`.
   - (The Sandbox - updated): Fixed in commit *478d4b8391e9aba2f7e13fb66b6abeaaa7b22473*.

8. 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.

**LandSale.sol** commit $eb2866621f04691b8037b1fca03d335ab29bb8e3$, previous

1. INFO `buyLand()`: 408 (size of the land) is hard coded. Recommend storing it as `constant`.

    - (The Sandbox - updated): Fixed in commit $328a3024d7100b7c645fc3e3338eb96896de852b$.

# 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 Admin.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 MetaTransactionReceiver.sol

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

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*

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

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

*Raw code*

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

| | |
|---|---|
| *Counterexample* | This code violates the specification |

*Initial environment*

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
53             balance: 0x0
54         }
55     }
56
```

*Post environment*

```
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

## Formal Verification Request 1

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

10, Dec 2019
33.75 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
0.51 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;
```

```
        _wallet = initialWalletAddress;
        _merkleRoot = merkleRoot;
        _expiryTime = expiryTime;
    }
```

✅ The code meets the specification.

## Formal Verification Request 3

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

📅 10, Dec 2019
⏱ 0.65 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
⏱ 2.1 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
⏱ 22.31 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
⏱ 0.47 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
0.47 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
3.82 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
2.6 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
131.51 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       );
```

```

        require(
            _sand.transferFrom(
                buyer,
                _wallet,
                price
            ),
            "sand transfer failed"
        );

        _land.mintQuad(to, size, x, y, "");
        emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);
    }
```

The code meets the specification.

## Formal Verification Request 11

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

10, Dec 2019
16.45 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            ),
```

```
        "sand transfer failed"
    );

    _land.mintQuad(to, size, x, y, "");
    emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);
}
```

The code meets the specification.

## Formal Verification Request 12

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

10, Dec 2019
16.9 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
17.59 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
             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 14

### getExpiryTime

10, Dec 2019
3.88 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
3.96 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
0.42 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
⏱ 0.32 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
0.34 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
```

```
        @post i == proof.length
       */
      for (uint256 i = 0; i < proof.length; i++) {
         bytes32 proofElement = proof[i];

         if (computedHash < proofElement) {
            computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
         } else {
            computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
         }
      }

      return computedHash == _merkleRoot;
   }
```

The code meets the specification.

## Formal Verification Request 20

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

10, Dec 2019
0.36 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

⏱ 0.4 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
⏱ 16.79 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

### getAdmin

10, Dec 2019
4.5 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 24

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

10, Dec 2019
13.05 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 25

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

10, Dec 2019
0.37 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 26

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

📅 10, Dec 2019
⏱ 0.36 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 27

**changeAdmin_requirement**

📅 10, Dec 2019
⏱ 0.8 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 28

### changeAdmin_change

10, Dec 2019
1.15 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 29

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

10, Dec 2019
24.22 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 30

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

10, Dec 2019
0.42 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 31

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

📅 10, Dec 2019
⏱ 0.42 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 32

**setMetaTransactionProcessor**

📅 10, Dec 2019
⏱ 0.85 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)
          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 33

**setMetaTransactionProcessor**

📅 10, Dec 2019
⏱ 0.85 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)
          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 34

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

📅 10, Dec 2019
⏱ 0.35 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 35

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

10, Dec 2019
0.38 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 36

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

10, Dec 2019
0.39 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 37

**_setMetaTransactionProcessor**

10, Dec 2019
1.12 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 38

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

📅 10, Dec 2019
⏱ 5.38 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 39

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

📅 10, Dec 2019
⏱ 0.3 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 40

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

10, Dec 2019
0.49 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 41

**isMetaTransactionProcessor**

10, Dec 2019
0.32 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 42

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

10, Dec 2019
23.89 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 {
```

```
        _admin = admin;
        _setMetaTransactionProcessor(metaTransactionContract, true);
    }
```

The code meets the specification.

## Formal Verification Request 43

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

10, Dec 2019
0.61 ms

Line 26 in File ERC721BaseToken.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 33-39 in File ERC721BaseToken.sol

```
    constructor(
        address metaTransactionContract,
        address admin
    ) internal {
        _admin = admin;
        _setMetaTransactionProcessor(metaTransactionContract, true);
    }
```

The code meets the specification.

## Formal Verification Request 44

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

10, Dec 2019
0.38 ms

Line 27 in File ERC721BaseToken.sol

```
    //@CTK NO_ASF
```

Line 33-39 in File ERC721BaseToken.sol

```
    constructor(
        address metaTransactionContract,
        address admin
    ) internal {
        _admin = admin;
        _setMetaTransactionProcessor(metaTransactionContract, true);
    }
```

The code meets the specification.

## Formal Verification Request 45

### ERC721BaseToken

10, Dec 2019
1.38 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 46

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

10, Dec 2019
86.12 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
```

```
      __has_buf_overflow = false
      __has_overflow = false
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
    }
    Other = {
      block = {
        "number": 0,
        "timestamp": 0
      }
    }
    Address_Map = [
      {
      "key": 0,
      "value": {
        "contract_name": "ERC721BaseToken",
        "balance": 0,
        "contract": {
          "_ERC721_RECEIVED": "AAAA",
          "_ERC721_BATCH_RECEIVED": "AAAA",
          "ERC165ID": "AAAA",
          "ERC721_MANDATORY_RECEIVER": "AAAA",
          "_numNFTPerAddress": [
            {
              "key": 18,
              "value": 0
            },
            {
              "key": 48,
              "value": 2
            },
            {
              "key": 192,
              "value": 0
            },
            {
              "key": 40,
              "value": 16
            },
            {
              "key": 66,
              "value": 4
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 65,
              "value": 1
            },
            {
              "key": 1,
```

```
              "value": 0
            },
            {
              "key": 8,
              "value": 16
            },
            {
              "key": 0,
              "value": 182
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 32,
              "value": 2
            },
            {
              "key": 2,
              "value": 2
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 255
            }
          ],
          "_owners": [
            {
              "key": 64,
              "value": 64
            },
            {
              "key": 128,
              "value": 8
            },
            {
              "key": 1,
              "value": 1
            },
            {
              "key": 8,
              "value": 4
            },
            {
              "key": 0,
              "value": 2
            },
            {
              "key": 16,
```

```
            "value": 1
          },
          {
            "key": 4,
            "value": 4
          },
          {
            "key": 32,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 64,
            "value": 16
          },
          {
            "key": 128,
            "value": 32
          },
          {
            "key": 0,
            "value": 160
          },
          {
            "key": 16,
            "value": 32
          },
          {
            "key": 2,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
```

```
          "_admin": 0,
          "_superOperators": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
      from = 1
      id = 0
      to = 0
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = true
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
  }
  Other = {
      block = {
        "number": 0,
        "timestamp": 0
      }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "ERC721BaseToken",
        "balance": 0,
        "contract": {
          "_ERC721_RECEIVED": "AAAA",
          "_ERC721_BATCH_RECEIVED": "AAAA",
          "ERC165ID": "AAAA",
          "ERC721_MANDATORY_RECEIVER": "AAAA",
          "_numNFTPerAddress": [
            {
```

```
            "key": 18,
            "value": 0
          },
          {
            "key": 64,
            "value": 0
          },
          {
            "key": 192,
            "value": 0
          },
          {
            "key": 40,
            "value": 16
          },
          {
            "key": 66,
            "value": 4
          },
          {
            "key": 128,
            "value": 0
          },
          {
            "key": 65,
            "value": 1
          },
          {
            "key": 8,
            "value": 16
          },
          {
            "key": 0,
            "value": 183
          },
          {
            "key": 16,
            "value": 0
          },
          {
            "key": 4,
            "value": 0
          },
          {
            "key": 32,
            "value": 2
          },
          {
            "key": 2,
            "value": 2
          },
          {
            "key": 48,
            "value": 2
          },
          {
            "key": "ALL_OTHERS",
            "value": 255
```

```
            }
          ],
          "_owners": [
            {
              "key": 64,
              "value": 64
            },
            {
              "key": 128,
              "value": 8
            },
            {
              "key": 1,
              "value": 1
            },
            {
              "key": 8,
              "value": 4
            },
            {
              "key": 16,
              "value": 1
            },
            {
              "key": 4,
              "value": 4
            },
            {
              "key": 32,
              "value": 8
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 64,
              "value": 16
            },
            {
              "key": 128,
              "value": 32
            },
            {
              "key": 0,
```

```
              "value": 160
            },
            {
              "key": 16,
              "value": 32
            },
            {
              "key": 2,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```


## Formal Verification Request 47

**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);
```

```
        emit Transfer(from, to, id);
    }
```

The code meets the specification.

## Formal Verification Request 48

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

10, Dec 2019
0.33 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 49

**_transferFrom**

10, Dec 2019
4.55 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 50

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

10, Dec 2019
11.94 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 51

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

10, Dec 2019
0.32 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 52

**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 53

### balanceOf_require

10, Dec 2019
0.31 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 54

### balanceOf_change

10, Dec 2019
1.07 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 55

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

10, Dec 2019
4.08 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 56

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

📅 10, Dec 2019
⏱ 0.29 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 57

**__ownerOf**

📅 10, Dec 2019
⏱ 0.28 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 58

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

📅 10, Dec 2019
⏱ 5.83 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,
          bool operatorEnabled) {
99        uint256 data = _owners[id];
100       owner = address(data);
101       operatorEnabled = (data / 2**255) == 1;
102   }
```

✅ The code meets the specification.

## Formal Verification Request 59

**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,
          bool operatorEnabled) {
99        uint256 data = _owners[id];
100       owner = address(data);
101       operatorEnabled = (data / 2**255) == 1;
102   }
```

✅ The code meets the specification.

## Formal Verification Request 60

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

📅 10, Dec 2019
⏱ 9.62 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.

```
Counter Example:
Before Execution:
    Input = {
        id = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        operatorEnabled = false
        owner = 0
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "AAAA",
            "ERC721_MANDATORY_RECEIVER": "CCCC",
            "_numNFTPerAddress": [
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
            "_owners": [
              {
                "key": 0,
                "value": 32
              },
              {
                "key": 2,
                "value": 34
              },
              {
                "key": 64,
                "value": 2
              },
              {
                "key": 128,
```

```
            "value": 16
          },
          {
            "key": 16,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 8
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": true
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 2,
            "value": 16
          },
          {
            "key": 128,
            "value": 128
          },
          {
            "key": 8,
            "value": 64
          },
          {
            "key": 64,
            "value": 4
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": "ALL_OTHERS",
            "value": true
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      }
```

```
      }
    }
  ]

Function invocation is reverted.
```

## Formal Verification Request 61

### _ownerAndOperatorEnabledOf

10, Dec 2019
0.37 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,
          bool operatorEnabled) {
99        uint256 data = _owners[id];
100       owner = address(data);
101       operatorEnabled = (data / 2**255) == 1;
102   }
```

The code meets the specification.

## Formal Verification Request 62

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

10, Dec 2019
20.05 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 63

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

10, Dec 2019
0.38 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 64

**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 65

**ownerOf**

10, Dec 2019
0.89 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 66

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

10, Dec 2019
16.45 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
                   overriden 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 67

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

10, Dec 2019
0.5 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
                   overriden 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 68

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

10, Dec 2019
0.33 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
                   overriden 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 69

**__approveFor**

10, Dec 2019
2.01 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
                   overriden 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 70

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

10, Dec 2019
61.45 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 71

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

📅 10, Dec 2019
⏱ 4.66 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 72

### Method will not encounter an assertion failure.

10, Dec 2019
4.7 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 73

### approveFor_require

10, Dec 2019
11.72 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]) || (
              _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])
154      */
```

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");
```

```
        require(
            msg.sender == sender ||
            _metaTransactionContracts[msg.sender] ||
            _superOperators[msg.sender] ||
            _operatorsForAll[sender][msg.sender],
            "not authorized to approve"
        );
        require(owner == sender, "owner != sender");
        _approveFor(owner, operator, id);
    }
```

✅ The code meets the specification.

## Formal Verification Request 74

### approveFor_change

📅 10, Dec 2019
⏱ 3.49 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]) || (
              _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])
160       @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
161       @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
              2**255)
162       @post (operator != address(0)) -> (__post._operators[id] == operator)
163      */
```

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 75

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

10, Dec 2019
53.63 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 76

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

10, Dec 2019
3.67 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 77

### Method will not encounter an assertion failure.

10, Dec 2019
3.59 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 78

### approve_require

10, Dec 2019
3.87 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]) || (
           _operatorsForAll[address(_owners[id])][msg.sender])
194     */
```

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 79

### approve_change

10, Dec 2019
3.26 ms

Line 195-202 in File ERC721BaseToken.sol

```
/*@CTK approve_change
  @tag assume_completion
  @pre address(_owners[id]) != address(0)
  @pre (msg.sender == address(_owners[id])) || (_superOperators[msg.sender]) || (
      _operatorsForAll[address(_owners[id])][msg.sender])
  @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
  @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
      2**255)
  @post (operator != address(0)) -> (__post._operators[id] == operator)
 */
```

Line 203-213 in File ERC721BaseToken.sol

```
function approve(address operator, uint256 id) external {
    address owner = _ownerOf(id);
    require(owner != address(0), "token does not exist");
    require(
        owner == msg.sender ||
        _superOperators[msg.sender] ||
        _operatorsForAll[owner][msg.sender],
        "not authorized to approve"
    );
    _approveFor(owner, operator, id);
}
```

The code meets the specification.

## Formal Verification Request 80

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

10, Dec 2019
27.01 ms

Line 220 in File ERC721BaseToken.sol

```
//@CTK NO_OVERFLOW
```

Line 229-237 in File ERC721BaseToken.sol

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

The code meets the specification.

## Formal Verification Request 81

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

10, Dec 2019
0.46 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 82

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

10, Dec 2019
9.58 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      }
```

```
This = 0
Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = false
    __has_returned = false
    __reverted = false
    msg = {
      "gas": 0,
      "sender": 0,
      "value": 0
    }
}
Other = {
    __return = 0
    block = {
      "number": 0,
      "timestamp": 0
    }
}
Address_Map = [
  {
    "key": "ALL_OTHERS",
    "value": {
      "contract_name": "ERC721BaseToken",
      "balance": 0,
      "contract": {
        "_ERC721_RECEIVED": "AAAA",
        "_ERC721_BATCH_RECEIVED": "AAAA",
        "ERC165ID": "AAAA",
        "ERC721_MANDATORY_RECEIVER": "\u0081\u0081\u0081\u0081",
        "_numNFTPerAddress": [
          {
            "key": 0,
            "value": 64
          },
          {
            "key": 4,
            "value": 8
          },
          {
            "key": 1,
            "value": 16
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_owners": [
          {
            "key": 128,
            "value": 16
          },
          {
            "key": 2,
            "value": 32
          },
```

```
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 16,
              "value": 128
            },
            {
              "key": 8,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ]
        }
      }
    }
  ]

Function invocation is reverted.
```


## Formal Verification Request 83

### getApproved

📅 10, Dec 2019
⏱ 0.46 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 84

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

10, Dec 2019
51.28 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 (
        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 85

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

10, Dec 2019
0.86 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 86

### Method will not encounter an assertion failure.

10, Dec 2019
10.79 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            );
```

```
        }
    }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 0
        id = 0
        to = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        isMetaTx = false
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "QQQQ",
            "ERC721_MANDATORY_RECEIVER": "AAAA",
            "_numNFTPerAddress": [
              {
                "key": 8,
                "value": 64
              },
              {
                "key": 4,
                "value": 8
              },
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
            "_owners": [
              {
```

```
            "key": 2,
            "value": 32
          },
          {
            "key": 16,
            "value": 2
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 4,
            "value": 64
          },
          {
            "key": 0,
            "value": 32
          },
          {
            "key": 64,
            "value": 4
          },
          {
            "key": 16,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": "ALL_OTHERS",
            "value": true
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
```

```
        }
      }
    }
  ]

Function invocation is reverted.
```

## Formal Verification Request 87

### _checkTransfer

10, Dec 2019
1.07 ms

Line 242-248 in File ERC721BaseToken.sol

```
  /*@CTK _checkTransfer
    @tag assume_completion
    @post address(_owners[id]) != address(0)
    @post from == _owners[id]
    @post to != address(0)
    @post (msg.sender != from) && (_metaTransactionContracts[msg.sender] == false) ->
        _superOperators[msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id
        ] / 2**255) == 1) && _operators[id] == msg.sender)
   */
```

Line 249-263 in File ERC721BaseToken.sol

```
  function _checkTransfer(address from, address to, uint256 id) internal view returns (
      bool isMetaTx) {
      (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
      require(owner != address(0), "token does not exist");
      require(owner == from, "not owner in _checkTransfer");
      require(to != address(0), "can't send to zero address");
      isMetaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
      if (msg.sender != from && !isMetaTx) {
          require(
              _superOperators[msg.sender] ||
              _operatorsForAll[from][msg.sender] ||
              (operatorEnabled && _operators[id] == msg.sender),
              "not approved to transfer"
          );
      }
  }
```

The code meets the specification.

## Formal Verification Request 88

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

10, Dec 2019
6.52 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 89

**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
                   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 90

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

📅 10, Dec 2019
⏱ 0.37 ms

Line 267 in File ERC721BaseToken.sol

```
267    //@CTK NO_ASF
```

Line 272-314 in File ERC721BaseToken.sol

```
    function _checkInterfaceWith10000Gas(address _contract, bytes4 interfaceId)
        internal
        view
        returns (bool)
    {
        bool success;
        bool result;
        bytes memory call_data = abi.encodeWithSelector(
            ERC165ID,
            interfaceId
        );
        // solium-disable-next-line security/no-inline-assembly
        /*@CTK _checkInterfaceWith10000Gas_assembly
          @tag assume_completion
          @var bool success
          @var bool result
          @post result == true
          @post success == true
         */
        // solium-disable-next-line security/no-inline-assembly
        assembly {
            let call_ptr := add(0x20, call_data)
            let call_size := mload(call_data)
            let output := mload(0x40) // Find empty storage location using "free memory
                pointer"
            mstore(output, 0x0)
            success := staticcall(
                10000,
                _contract,
                call_ptr,
                call_size,
                output,
                0x20
            ) // 32 bytes
            result := mload(output)
        }
        // (10000 / 63) "not enough for supportsInterface(...)" // consume all gas, so
            caller can potentially know that there was not enough gas
        assert(gasleft() > 158);
        return success && result;
    }
```

✅ The code meets the specification.

## Formal Verification Request 91

__checkInterfaceWith10000Gas

📅 10, Dec 2019
⏱ 0.43 ms

Line 268-271 in File ERC721BaseToken.sol

```
    /*@CTK _checkInterfaceWith10000Gas
      @tag assume_completion
      @post __return == true
     */
```

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 92

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

📅 10, Dec 2019
⏱ 77.43 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 93

**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 94

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

📅 10, Dec 2019
⏱ 10.7 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(
```

```
            _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, ""),
            "erc721 transfer rejected by to"
        );
    }
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 0
        id = 0
        to = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "EEEE",
            "ERC721_MANDATORY_RECEIVER": "AAAA",
            "_numNFTPerAddress": [
              {
                "key": 4,
                "value": 8
              },
              {
                "key": 8,
                "value": 64
              },
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
```

```
          "_owners": [
            {
              "key": 16,
              "value": 2
            },
            {
              "key": 2,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": true
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 16,
              "value": 128
            },
            {
              "key": 0,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ]
        }
      }
    }
  ]

Function invocation is reverted.
```

## Formal Verification Request 95

### transferFrom

10, Dec 2019
1.52 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 96

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

10, Dec 2019
76.83 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)
           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 97

**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)
            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 98

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

10, Dec 2019
9.66 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)
            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     }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        data = ""
        from = 0
        id = 0
        to = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "QQQQ",
            "ERC721_MANDATORY_RECEIVER": "AAAA",
            "_numNFTPerAddress": [
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 8,
                "value": 64
              },
              {
                "key": 64,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 8
              }
            ],
            "_owners": [
              {
```

```
            "key": 2,
            "value": 32
          },
          {
            "key": 16,
            "value": 2
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 0,
            "value": 32
          },
          {
            "key": 4,
            "value": 64
          },
          {
            "key": 64,
            "value": 4
          },
          {
            "key": 16,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": "ALL_OTHERS",
            "value": true
          }
        ]
```

```
        }
      }
    }
  ]

Function invocation is reverted.
```

## Formal Verification Request 99

### safeTransferFrom

📅 10, Dec 2019
⏱ 1.3 ms

Line 357-365 in File ERC721BaseToken.sol

```
  /*@CTK safeTransferFrom
    @tag assume_completion
    @pre (from == _owners[id]) && (from != address(0))
    @pre to != address(0)
    @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || _superOperators[
        msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id] / 2**255) ==
        1) && _operators[id] == msg.sender)
    @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
    @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
    @post __post._owners[id] == uint256(to)
   */
```

Line 366-377 in File ERC721BaseToken.sol

```
  function safeTransferFrom(address from, address to, uint256 id, bytes memory data)
      public {
      bool metaTx = _checkTransfer(from, to, id);
      _transferFrom(from, to, id);
      if (to.isContract()) {
          require(
              _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, data),
              "ERC721: transfer rejected by to"
          );
      }
  }
```

✅ The code meets the specification.

## Formal Verification Request 100

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

📅 10, Dec 2019
⏱ 5687.6 ms

Line 400 in File ERC721BaseToken.sol

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

Line 409-458 in File ERC721BaseToken.sol

```
function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory data, bool safe) internal {
    bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
    bool authorized = msg.sender == from ||
        metaTx ||
        _superOperators[msg.sender] ||
        _operatorsForAll[from][msg.sender];

    require(from != address(0), "from is zero address");
    require(to != address(0), "can't send to zero address");

    uint256 numTokens = ids.length;
    /*@CTK "_batchTransferFrom_loop"
      @pre from != address(0)
      @pre to != address(0)
      @pre numTokens < 5
      @inv this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @inv this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @inv ids == ids__pre
      @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from)
      @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))
      @inv i <= numTokens
      @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
      @inv numTokens == numTokens__pre
      @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre._numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
      @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @post i == numTokens
      @post !__should_return
     */
    for(uint256 i = 0; i < numTokens; i ++) {
        uint256 id = ids[i];
        (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
        require(owner == from, "not owner in batchTransferFrom");
        require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not authorized");
        _owners[id] = uint256(to);
        // emit Transfer(from, to, id);
    }
    if (from != to) {
        _numNFTPerAddress[from] -= numTokens;
        _numNFTPerAddress[to] += numTokens;
    }
    if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER))) {
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        data = ""
        from = 128
        ids = []
        safe = false
        to = 32
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "@@@@",
            "_ERC721_BATCH_RECEIVED": "@@@@",
            "ERC165ID": "@@@@",
            "ERC721_MANDATORY_RECEIVER": "@@@@",
            "_numNFTPerAddress": [
              {
                "key": 32,
                "value": 0
              },
              {
                "key": 0,
                "value": 128
              },
              {
                "key": 4,
                "value": 128
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
```

```
              "value": 255
            }
          ],
          "_owners": [
            {
              "key": 0,
              "value": 2
            },
            {
              "key": 128,
              "value": 8
            },
            {
              "key": 2,
              "value": 64
            },
            {
              "key": "ALL_OTHERS",
              "value": 255
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 8,
              "value": 0
            },
            {
              "key": 32,
              "value": 8
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 255
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
```

```
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
      data = ""
      from = 4
      ids = []
      safe = false
      to = 0
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = true
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
  }
  Other = {
      block = {
        "number": 0,
        "timestamp": 0
      }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "ERC721BaseToken",
        "balance": 0,
        "contract": {
          "_ERC721_RECEIVED": "@@@@",
          "_ERC721_BATCH_RECEIVED": "@@@@",
          "ERC165ID": "@@@@",
          "ERC721_MANDATORY_RECEIVER": "@@@@",
          "_numNFTPerAddress": [
            {
              "key": 32,
              "value": 0
            },
            {
```

```
            "key": 0,
            "value": 128
          },
          {
            "key": 4,
            "value": 128
          },
          {
            "key": 2,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 255
          }
        ],
        "_owners": [
          {
            "key": 0,
            "value": 2
          },
          {
            "key": 128,
            "value": 8
          },
          {
            "key": 2,
            "value": 64
          },
          {
            "key": "ALL_OTHERS",
            "value": 255
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 8,
            "value": 0
          },
          {
            "key": 32,
            "value": 8
          },
          {
            "key": 0,
            "value": 0
          },
```

```
            {
              "key": "ALL_OTHERS",
              "value": 255
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

## Formal Verification Request 101

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

📅 10, Dec 2019
⏱ 2.14 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]
```

```
        @inv ids == ids__pre
        @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from)
        @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))
        @inv i <= numTokens
        @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
        @inv numTokens == numTokens__pre
        @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre._numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
        @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
        @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
        @post i == numTokens
        @post !__should_return
       */
      for(uint256 i = 0; i < numTokens; i ++) {
         uint256 id = ids[i];
         (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
         require(owner == from, "not owner in batchTransferFrom");
         require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not authorized");
         _owners[id] = uint256(to);
         // emit Transfer(from, to, id);
      }
      if (from != to) {
         _numNFTPerAddress[from] -= numTokens;
         _numNFTPerAddress[to] += numTokens;
      }
      if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER))) {
         require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
         );
      }
   }
```

The code meets the specification.

## Formal Verification Request 102

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

10, Dec 2019
1.98 ms

Line 402 in File ERC721BaseToken.sol

```
   //@CTK NO_ASF
```

Line 409-458 in File ERC721BaseToken.sol

```
   function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory data, bool safe) internal {
      bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
      bool authorized = msg.sender == from ||
```

```
        metaTx ||
        _superOperators[msg.sender] ||
        _operatorsForAll[from][msg.sender];

    require(from != address(0), "from is zero address");
    require(to != address(0), "can't send to zero address");

    uint256 numTokens = ids.length;
    /*@CTK "_batchTransferFrom_loop"
      @pre from != address(0)
      @pre to != address(0)
      @pre numTokens < 5
      @inv this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @inv this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @inv ids == ids__pre
      @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from)
      @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))
      @inv i <= numTokens
      @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
      @inv numTokens == numTokens__pre
      @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre._numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
      @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @post i == numTokens
      @post !__should_return
     */
    for(uint256 i = 0; i < numTokens; i ++) {
        uint256 id = ids[i];
        (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
        require(owner == from, "not owner in batchTransferFrom");
        require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not authorized");
        _owners[id] = uint256(to);
        // emit Transfer(from, to, id);
    }
    if (from != to) {
        _numNFTPerAddress[from] -= numTokens;
        _numNFTPerAddress[to] += numTokens;
    }
    if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER))) {
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}
```

The code meets the specification.

# Formal Verification Request 103

## __batchTransferFrom

10, Dec 2019
3929.86 ms

Line 403-408 in File ERC721BaseToken.sol

```
/*@CTK FAIL "_batchTransferFrom"
  @tag assume_completion
  @pre from != address(0)
  @pre to != address(0)
  @post _numNFTPerAddress[from] + _numNFTPerAddress[to] == (__post._numNFTPerAddress[from] + __post._numNFTPerAddress[to])
 */
```

Line 409-458 in File ERC721BaseToken.sol

```
function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory data, bool safe) internal {
    bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
    bool authorized = msg.sender == from ||
        metaTx ||
        _superOperators[msg.sender] ||
        _operatorsForAll[from][msg.sender];

    require(from != address(0), "from is zero address");
    require(to != address(0), "can't send to zero address");

    uint256 numTokens = ids.length;
    /*@CTK "_batchTransferFrom_loop"
      @pre from != address(0)
      @pre to != address(0)
      @pre numTokens < 5
      @inv this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @inv this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @inv ids == ids__pre
      @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from)
      @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))
      @inv i <= numTokens
      @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
      @inv numTokens == numTokens__pre
      @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre._numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
      @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @post i == numTokens
      @post !__should_return
     */
    for(uint256 i = 0; i < numTokens; i ++) {
        uint256 id = ids[i];
        (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
        require(owner == from, "not owner in batchTransferFrom");
```

```
        require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not
            authorized");
        _owners[id] = uint256(to);
        // emit Transfer(from, to, id);
    }
    if (from != to) {
        _numNFTPerAddress[from] -= numTokens;
        _numNFTPerAddress[to] += numTokens;
    }
    if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to,
        ERC721_MANDATORY_RECEIVER))) {
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        data = ""
        from = 8
        ids = [
          16
        ]
        safe = false
        to = 8
    }
    This = 8
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC721Events",
          "balance": 0,
          "contract": {}
        }
      },
```

```
    {
      "key": 8,
      "value": {
        "contract_name": "ERC721BaseToken",
        "balance": 0,
        "contract": {
          "_ERC721_RECEIVED": "IIII",
          "_ERC721_BATCH_RECEIVED": "IIII",
          "ERC165ID": "IIII",
          "ERC721_MANDATORY_RECEIVER": "IIII",
          "_numNFTPerAddress": [
            {
              "key": 1,
              "value": 128
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 8
            }
          ],
          "_owners": [
            {
              "key": 8,
              "value": 7
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 1,
              "value": 128
            },
            {
              "key": 0,
              "value": 1
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 8
            }
          ],
```

```
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 1,
              "value": 128
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 8
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
    data = ""
```

```
        from = 8
        ids = [
          16
        ]
        safe = false
        to = 0
    }
    This = 8
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC721Events",
          "balance": 0,
          "contract": {}
        }
      },
      {
        "key": 8,
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "IIII",
            "_ERC721_BATCH_RECEIVED": "IIII",
            "ERC165ID": "IIII",
            "ERC721_MANDATORY_RECEIVER": "IIII",
            "_numNFTPerAddress": [
              {
                "key": 8,
                "value": 7
              },
              {
                "key": 128,
                "value": 0
              },
              {
                "key": 1,
                "value": 128
              },
```

```
            {
              "key": 0,
              "value": 1
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 8
            }
          ],
          "_owners": [
            {
              "key": 8,
              "value": 7
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 1,
              "value": 128
            },
            {
              "key": 0,
              "value": 1
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 8
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 1,
              "value": 128
            },
            {
              "key": 128,
              "value": 0
```

```
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 8
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

## Formal Verification Request 104

### supportsInterface

📅 10, Dec 2019
⏱ 10.29 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 105

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

10, Dec 2019
59.51 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 106

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

10, Dec 2019
5.53 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 107

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

10, Dec 2019
5.36 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 108

**__setApprovalForAll__require**

10, Dec 2019
6.72 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 {
```

```
        require(sender != address(0), "Invalid sender address");
        require(
            msg.sender == sender ||
            _metaTransactionContracts[msg.sender] ||
            _superOperators[msg.sender],
            "not authorized to approve for all"
        );

        _setApprovalForAll(sender, operator, approved);
    }
```

The code meets the specification.

## Formal Verification Request 109

**_setApprovalForAll_change**

10, Dec 2019
2.8 ms

Line 501-507 in File ERC721BaseToken.sol

```
    /*@CTK _setApprovalForAll_change
      @tag assume_completion
      @pre sender != address(0)
      @pre msg.sender == sender \/ _metaTransactionContracts[msg.sender] == true \/
          _superOperators[msg.sender] == true
      @pre _superOperators[operator] == false
      @post __post._operatorsForAll[sender][operator] == approved
     */
```

Line 508-522 in File ERC721BaseToken.sol

```
    function setApprovalForAllFor(
        address sender,
        address operator,
        bool approved
    ) external {
        require(sender != address(0), "Invalid sender address");
        require(
            msg.sender == sender ||
            _metaTransactionContracts[msg.sender] ||
            _superOperators[msg.sender],
            "not authorized to approve for all"
        );

        _setApprovalForAll(sender, operator, approved);
    }
```

The code meets the specification.

## Formal Verification Request 110

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

10, Dec 2019
22.1 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 111

**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 112

**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 113

### setApprovalForAll_require

10, Dec 2019
0.88 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 114

### setApprovalForAll_change

10, Dec 2019
1.83 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 115

### 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
568        emit ApprovalForAll(sender, operator, approved);
569    }
```

The code meets the specification.

## Formal Verification Request 116

**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
568        emit ApprovalForAll(sender, operator, approved);
569    }
```

The code meets the specification.

## Formal Verification Request 117

**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
568         emit ApprovalForAll(sender, operator, approved);
569     }
```

The code meets the specification.

## Formal Verification Request 118

__setApprovalForAll__require

10, Dec 2019
0.8 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
568         emit ApprovalForAll(sender, operator, approved);
569     }
```

The code meets the specification.

## Formal Verification Request 119

__setApprovalForAll__change

10, Dec 2019
1.86 ms

Line 552-556 in File ERC721BaseToken.sol

```
/*@CTK _setApprovalForAll_change
  @tag assume_completion
  @pre _superOperators[operator] == false
  @post __post._operatorsForAll[sender][operator] == approved
 */
```

Line 557-569 in File ERC721BaseToken.sol

```
function _setApprovalForAll(
    address sender,
    address operator,
    bool approved
) internal {
    require(
        !_superOperators[operator],
        "super operator can't have their approvalForAll changed"
    );
    _operatorsForAll[sender][operator] = approved;

    emit ApprovalForAll(sender, operator, approved);
}
```

✅ The code meets the specification.

## Formal Verification Request 120

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

📅 10, Dec 2019
⏱ 5.13 ms

Line 577 in File ERC721BaseToken.sol

```
//@CTK NO_OVERFLOW
```

Line 584-590 in File ERC721BaseToken.sol

```
function isApprovedForAll(address owner, address operator)
    external
    view
    returns (bool)
{
    return _operatorsForAll[owner][operator] || _superOperators[operator];
}
```

✅ The code meets the specification.

## Formal Verification Request 121

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

📅 10, Dec 2019
⏱ 0.26 ms

Line 578 in File ERC721BaseToken.sol

```
//@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 122

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

10, Dec 2019
0.39 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 123

**isApprovedForAll**

10, Dec 2019
1.06 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)
```

```
    {
        return _operatorsForAll[owner][operator] || _superOperators[operator];
    }
```

The code meets the specification.

## Formal Verification Request 124

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

10, Dec 2019
38.41 ms

Line 592 in File ERC721BaseToken.sol

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

Line 605-610 in File ERC721BaseToken.sol

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

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 0
        id = 0
        owner = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
```

```
        "contract_name": "ERC721BaseToken",
        "balance": 0,
        "contract": {
          "_ERC721_RECEIVED": "AAAA",
          "_ERC721_BATCH_RECEIVED": "AAAA",
          "ERC165ID": "AAAA",
          "ERC721_MANDATORY_RECEIVER": "AAAA",
          "_numNFTPerAddress": [
            {
              "key": 64,
              "value": 1
            },
            {
              "key": 8,
              "value": 2
            },
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 130,
              "value": 0
            },
            {
              "key": 40,
              "value": 0
            },
            {
              "key": 1,
              "value": 4
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 80,
              "value": 0
            },
            {
              "key": 128,
              "value": 8
            },
            {
              "key": "ALL_OTHERS",
              "value": 255
            }
          ],
          "_owners": [
            {
              "key": 8,
              "value": 16
            },
```

```
            {
              "key": 4,
              "value": 2
            },
            {
              "key": 0,
              "value": 1
            },
            {
              "key": 80,
              "value": 16
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 4,
              "value": 128
            },
            {
              "key": 16,
              "value": 64
            },
            {
              "key": 128,
              "value": 16
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
```

```
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
      from = 0
      id = 0
      owner = 0
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = true
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
  }
  Other = {
      block = {
        "number": 0,
        "timestamp": 0
      }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "ERC721BaseToken",
        "balance": 0,
        "contract": {
          "_ERC721_RECEIVED": "AAAA",
          "_ERC721_BATCH_RECEIVED": "AAAA",
          "ERC165ID": "AAAA",
          "ERC721_MANDATORY_RECEIVER": "AAAA",
          "_numNFTPerAddress": [
            {
              "key": 64,
              "value": 1
            },
            {
              "key": 8,
              "value": 2
```

```
            },
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 130,
              "value": 0
            },
            {
              "key": 40,
              "value": 0
            },
            {
              "key": 1,
              "value": 4
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 128,
              "value": 8
            },
            {
              "key": 80,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 255
            }
          ],
          "_owners": [
            {
              "key": 8,
              "value": 16
            },
            {
              "key": 4,
              "value": 2
            },
            {
              "key": 80,
              "value": 16
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
```

```
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 4,
            "value": 128
          },
          {
            "key": 16,
            "value": 64
          },
          {
            "key": 128,
            "value": 16
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 0,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      }
    }
  },
  {
    "key": "ALL_OTHERS",
    "value": "EmptyAddress"
  }
]
```


## Formal Verification Request 125

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

📅 10, Dec 2019
⏱ 0.45 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 126

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

📅 10, Dec 2019
⏱ 0.38 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 127

**__burn__require**

📅 10, Dec 2019
⏱ 0.42 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 128

**_burn_change**

10, Dec 2019
1.65 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 129

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

10, Dec 2019
53.69 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
```

```
      __has_overflow = false
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
    }
    Other = {
      block = {
        "number": 0,
        "timestamp": 0
      }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "AAAA",
            "ERC721_MANDATORY_RECEIVER": "AAAA",
            "_numNFTPerAddress": [
              {
                "key": 129,
                "value": 4
              },
              {
                "key": 32,
                "value": 2
              },
              {
                "key": 33,
                "value": 4
              },
              {
                "key": 1,
                "value": 1
              },
              {
                "key": 4,
                "value": 2
              },
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
            "_owners": [
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
```

```
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 2,
              "value": 64
            },
            {
              "key": 4,
              "value": 128
            },
            {
              "key": 0,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
    Input = {
        id = 0
```

```
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ERC721BaseToken",
          "balance": 0,
          "contract": {
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "AAAA",
            "ERC721_MANDATORY_RECEIVER": "AAAA",
            "_numNFTPerAddress": [
              {
                "key": 32,
                "value": 2
              },
              {
                "key": 0,
                "value": 255
              },
              {
                "key": 129,
                "value": 4
              },
              {
                "key": 33,
                "value": 4
              },
              {
                "key": 4,
                "value": 2
              },
              {
                "key": 1,
                "value": 1
              },
              {
                "key": "ALL_OTHERS",
```

```
              "value": 0
            }
          ],
          "_owners": [
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 2,
              "value": 64
            },
            {
              "key": 4,
              "value": 128
            },
            {
              "key": 0,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
```

```
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

## Formal Verification Request 130

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

10, Dec 2019
0.57 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 131

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

10, Dec 2019
0.51 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 132

**burn_require**

10, Dec 2019
0.98 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 133

burn_change

10, Dec 2019
6.0 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 134

If method completes, integer overflow would not happen.

10, Dec 2019
61.52 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 135

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

10, Dec 2019
8.69 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 136

**burnFrom_require**

10, Dec 2019
4.17 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] ||
```

```
        (operatorEnabled && _operators[id] == msg.sender) ||
        _superOperators[msg.sender] ||
        _operatorsForAll[from][msg.sender],
        "not authorized to burn"
    );
    _burn(from, owner, id);
}
```

The code meets the specification.

## Formal Verification Request 137

burnFrom_change

10, Dec 2019
3.27 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 138

__batchTransferFrom__loop___Generated

10, Dec 2019
81.15 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 139

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

10, Dec 2019
13.86 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 140

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

10, Dec 2019
0.4 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 141

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

10, Dec 2019
0.32 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 142

### setSuperOperator_admin

📅 10, Dec 2019
⏱ 0.24 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 143

### setSuperOperator_change

📅 10, Dec 2019
⏱ 1.35 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 144

**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 145

**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 146

**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 147

**isSuperOperator**

10, Dec 2019
0.27 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 148

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

10, Dec 2019
3.83 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 149

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

10, Dec 2019
0.4 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 150

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

10, Dec 2019
0.32 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 151

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

10, Dec 2019
112.9 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 152

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

10, Dec 2019
0.58 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 153

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

10, Dec 2019
0.55 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 154

**Land**

10, Dec 2019
2.66 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 155

**name**

📅 10, Dec 2019
⏱ 5.5 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 156

**symbol**

📅 10, Dec 2019
⏱ 4.4 ms

Line 40-42 in File Land.sol

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

Line 43-45 in File Land.sol

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

The code meets the specification.

## Formal Verification Request 157

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

10, Dec 2019
7.34 ms

Line 93 in File Land.sol

```
    //@CTK NO_OVERFLOW
```

Line 101-103 in File Land.sol

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

The code meets the specification.

## Formal Verification Request 158

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

10, Dec 2019
0.27 ms

Line 94 in File Land.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 101-103 in File Land.sol

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

The code meets the specification.

## Formal Verification Request 159

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

10, Dec 2019
0.27 ms

Line 95 in File Land.sol

```
    //@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 160

### supportsInterface

10, Dec 2019
1.44 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 161

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

10, Dec 2019
15.64 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 162

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

10, Dec 2019
0.42 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 163

**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 164

**setMinter_require**

10, Dec 2019
1.27 ms

Line 26-29 in File LandBaseToken.sol

```
/*@CTK setMinter_require
  @tag assume_completion
  @post msg.sender == _admin
 */
```

Line 34-41 in File LandBaseToken.sol

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

✅ The code meets the specification.

## Formal Verification Request 165

**setMinter_change**

📅 10, Dec 2019
⏱ 4.12 ms

Line 30-33 in File LandBaseToken.sol

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

Line 34-41 in File LandBaseToken.sol

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

✅ The code meets the specification.

## Formal Verification Request 166

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

📅 10, Dec 2019
⏱ 4.77 ms

Line 46 in File LandBaseToken.sol

```
//@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 167

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

10, Dec 2019
0.32 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 168

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

10, Dec 2019
0.31 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 169

**isMinter**

10, Dec 2019
0.4 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 170

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

10, Dec 2019
58.69 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 171

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

10, Dec 2019
0.49 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 172

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

10, Dec 2019
0.48 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 173

### LandBaseToken

📅 10, Dec 2019
⏱ 1.77 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 174

### width

📅 10, Dec 2019
⏱ 3.81 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 175

**height**

10, Dec 2019
3.76 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 176

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

10, Dec 2019
126.29 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 177

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

10, Dec 2019
0.95 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 178

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

10, Dec 2019
1.05 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 179

**x**

10, Dec 2019
0.98 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 180

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

10, Dec 2019
140.79 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 181

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

10, Dec 2019
0.97 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 182

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

10, Dec 2019
1.05 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 183

**y**

10, Dec 2019
1.02 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 184

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

📅 10, Dec 2019
⏱ 480.9 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");
```

```
        uint256 toX = x+size;
        uint256 toY = y+size;
        if (size <= 12) {
            require(
                _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
                "Already minted as 12x12"
            );
        } else {
            /*@*CTK mintQuad_loop1
              @tag assume_completion
              @inv x12i <= x + size
              @post x12i == x + size
             */
            for (uint256 x12i = x; x12i < toX; x12i += 12) {
                for (uint256 y12i = y; y12i < toY; y12i += 12) {
                    uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
                    require(_owners[id12x12] == 0, "Already minted as 12x12");
                }
            }
        }

        if (size <= 6) {
            require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
                minted as 6x6");
        } else {
            for (uint256 x6i = x; x6i < toX; x6i += 6) {
                for (uint256 y6i = y; y6i < toY; y6i += 6) {
                    uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
                    require(_owners[id6x6] == 0, "Already minted as 6x6");
                }
            }
        }

        if (size <= 3) {
            require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
                minted as 3x3");
        } else {
            for (uint256 x3i = x; x3i < toX; x3i += 3) {
                for (uint256 y3i = y; y3i < toY; y3i += 3) {
                    uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
                    require(_owners[id3x3] == 0, "Already minted as 3x3");
                }
            }
        }

        /*@*CTK mintQuad_loopx
          @tag assume_completion
          @pre GRID_SIZE == 408
          @inv i <= size * size
          @post i == size * size
         */
        for (uint256 i = 0; i < size*size; i++) {
            uint256 id = _idInPath(i, size, x, y);
            require(_owners[id] == 0, "Already minted");
            emit Transfer(address(0), to, id);
        }

```

```
        _owners[quadId] = uint256(to);
        _numNFTPerAddress[to] += size * size;

        _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
    }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        data = ""
        size = 12
        to = 128
        x = 0
        y = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 64,
            "LAYER": 0,
            "LAYER_1x1": 32,
            "LAYER_3x3": 0,
            "LAYER_6x6": 2,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": 8,
                "value": true
              },
              {
                "key": 0,
                "value": true
              },
```

```
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_ERC721_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
          "_ERC721_BATCH_RECEIVED": "GGGG",
          "ERC165ID": "\u00c1\u00c1\u00c1\u00c1",
          "ERC721_MANDATORY_RECEIVER": "GGGG",
          "_numNFTPerAddress": [
            {
              "key": 0,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 128
            }
          ],
          "_owners": [
            {
              "key": 1,
              "value": 0
            },
            {
              "key": 80,
              "value": 0
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 128
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 1,
              "value": 0
            },
            {
```

```
              "key": 80,
              "value": 0
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 128
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 2,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
    data = ""
    size = 12
    to = 128
    x = 0
    y = 0
  }
  This = 0
  Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
```

```
      __has_overflow = true
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
    }
    Other = {
      block = {
        "number": 0,
        "timestamp": 0
      }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 64,
            "LAYER": 0,
            "LAYER_1x1": 32,
            "LAYER_3x3": 0,
            "LAYER_6x6": 2,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": 8,
                "value": true
              },
              {
                "key": 0,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "_ERC721_RECEIVED": "\u00c1\u00c1\u00c1\u00c1",
            "_ERC721_BATCH_RECEIVED": "GGGG",
            "ERC165ID": "\u00c1\u00c1\u00c1\u00c1",
            "ERC721_MANDATORY_RECEIVER": "GGGG",
            "_numNFTPerAddress": [
              {
                "key": 0,
                "value": 4
              },
              {
                "key": 128,
                "value": 16
              },
              {
                "key": "ALL_OTHERS",
```

```
            "value": 128
          }
        ],
        "_owners": [
          {
            "key": 1,
            "value": 0
          },
          {
            "key": 80,
            "value": 0
          },
          {
            "key": 128,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 128
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 1,
            "value": 0
          },
          {
            "key": 80,
            "value": 0
          },
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 128,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 128
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 0,
            "value": true
```

```
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 2,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

## Formal Verification Request 185

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

10, Dec 2019
58.79 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)
        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;
```

```
        } else if (size == 12) {
            quadId = LAYER_12x12 + id;
        } else if (size == 24) {
            quadId = LAYER_24x24 + id;
        } else {
            require(false, "Invalid size");
        }

        require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
            Already minted as 24x24");

        uint256 toX = x+size;
        uint256 toY = y+size;
        if (size <= 12) {
            require(
                _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
                "Already minted as 12x12"
            );
        } else {
            /*@*CTK mintQuad_loop1
              @tag assume_completion
              @inv x12i <= x + size
              @post x12i == x + size
             */
            for (uint256 x12i = x; x12i < toX; x12i += 12) {
                for (uint256 y12i = y; y12i < toY; y12i += 12) {
                    uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
                    require(_owners[id12x12] == 0, "Already minted as 12x12");
                }
            }
        }

        if (size <= 6) {
            require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
                minted as 6x6");
        } else {
            for (uint256 x6i = x; x6i < toX; x6i += 6) {
                for (uint256 y6i = y; y6i < toY; y6i += 6) {
                    uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
                    require(_owners[id6x6] == 0, "Already minted as 6x6");
                }
            }
        }

        if (size <= 3) {
            require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
                minted as 3x3");
        } else {
            for (uint256 x3i = x; x3i < toX; x3i += 3) {
                for (uint256 y3i = y; y3i < toY; y3i += 3) {
                    uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
                    require(_owners[id3x3] == 0, "Already minted as 3x3");
                }
            }
        }

        /*@*CTK mintQuad_loopx
          @tag assume_completion
```

```
        @pre GRID_SIZE == 408
        @inv i <= size * size
        @post i == size * size
       */
      for (uint256 i = 0; i < size*size; i++) {
         uint256 id = _idInPath(i, size, x, y);
         require(_owners[id] == 0, "Already minted");
         emit Transfer(address(0), to, id);
      }

      _owners[quadId] = uint256(to);
      _numNFTPerAddress[to] += size * size;

      _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
   }
```

✅ The code meets the specification.

## Formal Verification Request 186

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

📅 10, Dec 2019
⏱ 69.79 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)
         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
```

```
        require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
            Already minted as 24x24");

        uint256 toX = x+size;
        uint256 toY = y+size;
        if (size <= 12) {
            require(
                _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
                "Already minted as 12x12"
            );
        } else {
            /*@*CTK mintQuad_loop1
              @tag assume_completion
              @inv x12i <= x + size
              @post x12i == x + size
             */
            for (uint256 x12i = x; x12i < toX; x12i += 12) {
                for (uint256 y12i = y; y12i < toY; y12i += 12) {
                    uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
                    require(_owners[id12x12] == 0, "Already minted as 12x12");
                }
            }
        }

        if (size <= 6) {
            require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
                minted as 6x6");
        } else {
            for (uint256 x6i = x; x6i < toX; x6i += 6) {
                for (uint256 y6i = y; y6i < toY; y6i += 6) {
                    uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
                    require(_owners[id6x6] == 0, "Already minted as 6x6");
                }
            }
        }

        if (size <= 3) {
            require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
                minted as 3x3");
        } else {
            for (uint256 x3i = x; x3i < toX; x3i += 3) {
                for (uint256 y3i = y; y3i < toY; y3i += 3) {
                    uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
                    require(_owners[id3x3] == 0, "Already minted as 3x3");
                }
            }
        }

        /*@*CTK mintQuad_loopx
          @tag assume_completion
          @pre GRID_SIZE == 408
          @inv i <= size * size
          @post i == size * size
         */
        for (uint256 i = 0; i < size*size; i++) {
            uint256 id = _idInPath(i, size, x, y);
            require(_owners[id] == 0, "Already minted");
            emit Transfer(address(0), to, id);
```

```
    }

    _owners[quadId] = uint256(to);
    _numNFTPerAddress[to] += size * size;

    _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
}
```

The code meets the specification.

## Formal Verification Request 187

### mintQuad_require

10, Dec 2019
2632.29 ms

Line 134-146 in File LandBaseToken.sol

```
/*@CTK mintQuad_require
  @tag assume_completion
  @pre GRID_SIZE == 408
  @post to != address(0)
  @post _minters[msg.sender] == true
  @post (x % size == 0) /\ (y % size == 0)
  @post (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
  @post (size == 1 \/ size == 3 \/ size == 6 \/ size == 12 \/ size == 24)
  @post _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0
  @post size <= 12 -> _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] ==
      0
  @post size <= 6 -> _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0
  @post size <= 3 -> _owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0
 */
```

Line 167-264 in File LandBaseToken.sol

```
function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data)
    external {
    require(to != address(0), "to is zero address");
    require(
        isMinter(msg.sender),
        "Only a minter can mint"
    );
    require(x % size == 0 && y % size == 0, "Invalid coordinates");
    require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");

    uint256 quadId;
    uint256 id = x + y * GRID_SIZE;

    if (size == 1) {
        quadId = id;
    } else if (size == 3) {
        quadId = LAYER_3x3 + id;
    } else if (size == 6) {
        quadId = LAYER_6x6 + id;
    } else if (size == 12) {
        quadId = LAYER_12x12 + id;
    } else if (size == 24) {
```

```
            quadId = LAYER_24x24 + id;
        } else {
            require(false, "Invalid size");
        }

        require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
            Already minted as 24x24");

        uint256 toX = x+size;
        uint256 toY = y+size;
        if (size <= 12) {
            require(
                _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
                "Already minted as 12x12"
            );
        } else {
            /*@*CTK mintQuad_loop1
              @tag assume_completion
              @inv x12i <= x + size
              @post x12i == x + size
             */
            for (uint256 x12i = x; x12i < toX; x12i += 12) {
                for (uint256 y12i = y; y12i < toY; y12i += 12) {
                    uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
                    require(_owners[id12x12] == 0, "Already minted as 12x12");
                }
            }
        }

        if (size <= 6) {
            require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
                minted as 6x6");
        } else {
            for (uint256 x6i = x; x6i < toX; x6i += 6) {
                for (uint256 y6i = y; y6i < toY; y6i += 6) {
                    uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
                    require(_owners[id6x6] == 0, "Already minted as 6x6");
                }
            }
        }

        if (size <= 3) {
            require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
                minted as 3x3");
        } else {
            for (uint256 x3i = x; x3i < toX; x3i += 3) {
                for (uint256 y3i = y; y3i < toY; y3i += 3) {
                    uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
                    require(_owners[id3x3] == 0, "Already minted as 3x3");
                }
            }
        }

        /*@*CTK mintQuad_loopx
          @tag assume_completion
          @pre GRID_SIZE == 408
          @inv i <= size * size
          @post i == size * size
```

```
        */
        for (uint256 i = 0; i < size*size; i++) {
            uint256 id = _idInPath(i, size, x, y);
            require(_owners[id] == 0, "Already minted");
            emit Transfer(address(0), to, id);
        }

        _owners[quadId] = uint256(to);
        _numNFTPerAddress[to] += size * size;

        _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
    }
```

✅ The code meets the specification.

## Formal Verification Request 188

### mintQuad_change

📅 10, Dec 2019
⏱ 230.24 ms

Line 147-166 in File LandBaseToken.sol

```
    /*@CTK mintQuad_change
      @tag assume_completion
      @pre GRID_SIZE == 408
      @pre LAYER == 0xFF0000000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_1x1 == 0x0000000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_3x3 == 0x0100000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_6x6 == 0x0200000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_12x12 == 0x0300000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_24x24 == 0x0400000000000000000000000000000000000000000000000000000000000000
      @pre to != address(0)
      @pre _minters[msg.sender] == true
      @pre (x % size == 0) /\ (y % size == 0)
      @pre (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
      @post (size == 1 \/ size == 3 \/ size == 6 \/ size == 12 \/ size == 24)
      @pre _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0
      @pre size <= 12 -> _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0
      @pre size <= 6 -> _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0
      @pre size <= 3 -> _owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0
      @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + (size * size)
     */
```

Line 167-264 in File LandBaseToken.sol

```
    function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data)
        external {
        require(to != address(0), "to is zero address");
        require(
            isMinter(msg.sender),
            "Only a minter can mint"
        );
        require(x % size == 0 && y % size == 0, "Invalid coordinates");
        require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");

        uint256 quadId;
```

```
        uint256 id = x + y * GRID_SIZE;

        if (size == 1) {
            quadId = id;
        } else if (size == 3) {
            quadId = LAYER_3x3 + id;
        } else if (size == 6) {
            quadId = LAYER_6x6 + id;
        } else if (size == 12) {
            quadId = LAYER_12x12 + id;
        } else if (size == 24) {
            quadId = LAYER_24x24 + id;
        } else {
            require(false, "Invalid size");
        }

        require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "
            Already minted as 24x24");

        uint256 toX = x+size;
        uint256 toY = y+size;
        if (size <= 12) {
            require(
                _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
                "Already minted as 12x12"
            );
        } else {
            /*@*CTK mintQuad_loop1
              @tag assume_completion
              @inv x12i <= x + size
              @post x12i == x + size
             */
            for (uint256 x12i = x; x12i < toX; x12i += 12) {
                for (uint256 y12i = y; y12i < toY; y12i += 12) {
                    uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
                    require(_owners[id12x12] == 0, "Already minted as 12x12");
                }
            }
        }

        if (size <= 6) {
            require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already
                minted as 6x6");
        } else {
            for (uint256 x6i = x; x6i < toX; x6i += 6) {
                for (uint256 y6i = y; y6i < toY; y6i += 6) {
                    uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
                    require(_owners[id6x6] == 0, "Already minted as 6x6");
                }
            }
        }

        if (size <= 3) {
            require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
                minted as 3x3");
        } else {
            for (uint256 x3i = x; x3i < toX; x3i += 3) {
                for (uint256 y3i = y; y3i < toY; y3i += 3) {
```

```
                uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
                require(_owners[id3x3] == 0, "Already minted as 3x3");
            }
        }
    }

    /*@*CTK mintQuad_loopx
      @tag assume_completion
      @pre GRID_SIZE == 408
      @inv i <= size * size
      @post i == size * size
     */
    for (uint256 i = 0; i < size*size; i++) {
        uint256 id = _idInPath(i, size, x, y);
        require(_owners[id] == 0, "Already minted");
        emit Transfer(address(0), to, id);
    }

    _owners[quadId] = uint256(to);
    _numNFTPerAddress[to] += size * size;

    _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
}
```

The code meets the specification.

## Formal Verification Request 189

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

10, Dec 2019
36.83 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(
        uint256) {
276     uint256 row = i / size;
277     if(row % 2 == 0) { // alow 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.

```
Counter Example:
Before Execution:
    Input = {
        i = 32
        size = 130
        x = 32
        y = 50
```

```
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 18,
            "LAYER": 0,
            "LAYER_1x1": 0,
            "LAYER_3x3": 0,
            "LAYER_6x6": 0,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": 0,
                "value": false
              },
              {
                "key": "ALL_OTHERS",
                "value": true
              }
            ],
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "AAAA",
            "ERC721_MANDATORY_RECEIVER": "AAAA",
            "_numNFTPerAddress": [
              {
                "key": 64,
                "value": 2
              },
              {
                "key": 0,
                "value": 0
              },
```

```
            {
              "key": 4,
              "value": 2
            },
            {
              "key": "ALL_OTHERS",
              "value": 16
            }
          ],
          "_owners": [
            {
              "key": 0,
              "value": 1
            },
            {
              "key": 8,
              "value": 128
            },
            {
              "key": 2,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": 0,
                  "value": false
                },
                {
                  "key": "ALL_OTHERS",
                  "value": true
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 32,
              "value": 16
            },
            {
              "key": 0,
              "value": 8
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
```

```
              "key": "ALL_OTHERS",
              "value": true
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    }
  ]

After Execution:
  Input = {
      i = 32
      size = 130
      x = 32
      y = 50
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = true
      __has_returned = true
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
  }
  Other = {
      __return = 196
      block = {
        "number": 0,
        "timestamp": 0
      }
  }
  Address_Map = [
    {
      "key": "ALL_OTHERS",
      "value": {
        "contract_name": "LandBaseToken",
        "balance": 0,
        "contract": {
          "GRID_SIZE": 18,
          "LAYER": 0,
          "LAYER_1x1": 0,
          "LAYER_3x3": 0,
          "LAYER_6x6": 0,
          "LAYER_12x12": 0,
          "LAYER_24x24": 0,
          "_minters": [
```

```
            {
              "key": 0,
              "value": false
            },
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ],
          "_ERC721_RECEIVED": "AAAA",
          "_ERC721_BATCH_RECEIVED": "AAAA",
          "ERC165ID": "AAAA",
          "ERC721_MANDATORY_RECEIVER": "AAAA",
          "_numNFTPerAddress": [
            {
              "key": 64,
              "value": 2
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 4,
              "value": 2
            },
            {
              "key": "ALL_OTHERS",
              "value": 16
            }
          ],
          "_owners": [
            {
              "key": 0,
              "value": 1
            },
            {
              "key": 8,
              "value": 128
            },
            {
              "key": 2,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": 0,
                  "value": false
                },
                {
```

```
                "key": "ALL_OTHERS",
                "value": true
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 32,
            "value": 16
          },
          {
            "key": 0,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": "ALL_OTHERS",
            "value": true
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      }
    }
  }
]
```

## Formal Verification Request 190

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

📅 10, Dec 2019
⏱ 0.65 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(
        uint256) {
276     uint256 row = i / size;
277     if(row % 2 == 0) { // alow 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);
```

```
        }
    }
```

The code meets the specification.

## Formal Verification Request 191

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

10, Dec 2019
10.32 ms

Line 268 in File LandBaseToken.sol

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

Line 275-282 in File LandBaseToken.sol

```
    function _idInPath(uint256 i, uint256 size, uint256 x, uint256 y) internal pure returns(
        uint256) {
        uint256 row = i / size;
        if(row % 2 == 0) { // alow ids to follow a path in a quad
            return (x + (i%size)) + ((y + row) * GRID_SIZE);
        } else {
            return ((x + size) - (1 + i%size)) + ((y + row) * GRID_SIZE);
        }
    }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        i = 0
        size = 0
        x = 0
        y = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
```

```
Address_Map = [
  {
    "key": "ALL_OTHERS",
    "value": {
      "contract_name": "LandBaseToken",
      "balance": 0,
      "contract": {
        "GRID_SIZE": 0,
        "LAYER": 0,
        "LAYER_1x1": 0,
        "LAYER_3x3": 0,
        "LAYER_6x6": 0,
        "LAYER_12x12": 0,
        "LAYER_24x24": 0,
        "_minters": [
          {
            "key": 0,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_ERC721_RECEIVED": "AAAA",
        "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
        "ERC165ID": "AAAA",
        "ERC721_MANDATORY_RECEIVER": "AAAA",
        "_numNFTPerAddress": [
          {
            "key": 64,
            "value": 32
          },
          {
            "key": 32,
            "value": 64
          },
          {
            "key": 0,
            "value": 128
          },
          {
            "key": 2,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 2
          }
        ],
        "_owners": [
          {
            "key": 0,
            "value": 32
          },
          {
            "key": 16,
            "value": 128
```

```
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_operatorsForAll": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 2,
              "value": 1
            },
            {
              "key": 16,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ]
        }
      }
    }
  ]

Function invocation is reverted.
```


## Formal Verification Request 192

### _idInPath

📅 10, Dec 2019
⏱ 4.89 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) { // alow 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 193

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

📅 10, Dec 2019
⏱ 378.03 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.

```
Counter Example:
Before Execution:
    Input = {
        data = ""
        from = 64
        size = 60
        to = 1
        x = 0
        y = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 0,
            "LAYER": 0,
            "LAYER_1x1": 0,
            "LAYER_3x3": 0,
            "LAYER_6x6": 0,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": 128,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "_ERC721_RECEIVED": "}}}}",
            "_ERC721_BATCH_RECEIVED": "}}}}",
            "ERC165ID": "}}}}",
            "ERC721_MANDATORY_RECEIVER": "}}}}",
```

```
        "_numNFTPerAddress": [
          {
            "key": 65,
            "value": 0
          },
          {
            "key": 2,
            "value": 0
          },
          {
            "key": 0,
            "value": 128
          },
          {
            "key": 16,
            "value": 2
          },
          {
            "key": 128,
            "value": 0
          },
          {
            "key": 64,
            "value": 0
          },
          {
            "key": 1,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 60
          }
        ],
        "_owners": [
          {
            "key": 2,
            "value": 64
          },
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 1,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 60
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
```

```
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 2,
            "value": 64
          },
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 1,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 60
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 0,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": 2,
            "value": true
          },
          {
            "key": 16,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      }
    }
  },
  {
    "key": "ALL_OTHERS",
    "value": "EmptyAddress"
  }
]

After Execution:
```

```
Input = {
    data = ""
    from = 64
    size = 60
    to = 1
    x = 0
    y = 0
}
This = 0
Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = true
    __has_returned = false
    __reverted = false
    msg = {
      "gas": 0,
      "sender": 0,
      "value": 0
    }
}
Other = {
    block = {
      "number": 0,
      "timestamp": 0
    }
}
Address_Map = [
  {
    "key": 0,
    "value": {
      "contract_name": "LandBaseToken",
      "balance": 0,
      "contract": {
        "GRID_SIZE": 0,
        "LAYER": 0,
        "LAYER_1x1": 0,
        "LAYER_3x3": 0,
        "LAYER_6x6": 0,
        "LAYER_12x12": 0,
        "LAYER_24x24": 0,
        "_minters": [
          {
            "key": 128,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_ERC721_RECEIVED": "}}}}",
        "_ERC721_BATCH_RECEIVED": "}}}}",
        "ERC165ID": "}}}}",
        "ERC721_MANDATORY_RECEIVER": "}}}}",
        "_numNFTPerAddress": [
          {
            "key": 16,
```

```
            "value": 2
          },
          {
            "key": 2,
            "value": 0
          },
          {
            "key": 0,
            "value": 128
          },
          {
            "key": 65,
            "value": 0
          },
          {
            "key": 128,
            "value": 0
          },
          {
            "key": 64,
            "value": 240
          },
          {
            "key": 1,
            "value": 16
          },
          {
            "key": "ALL_OTHERS",
            "value": 60
          }
        ],
        "_owners": [
          {
            "key": 2,
            "value": 64
          },
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 1,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 60
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
```

```
            }
          ],
          "_operators": [
            {
              "key": 2,
              "value": 64
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 1,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 60
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 2,
              "value": true
            },
            {
              "key": 16,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

## Formal Verification Request 194

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

10, Dec 2019
15.52 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 195

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

10, Dec 2019
11.49 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            );
```

```
        }
        _transferQuad(from, to, size, x, y);
        _numNFTPerAddress[from] -= size * size;
        _numNFTPerAddress[to] += size * size;

        _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data);
    }
```

The code meets the specification.

## Formal Verification Request 196

### transferQuad_require

10, Dec 2019
8.8 ms

Line 294-299 in File LandBaseToken.sol

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

Line 309-329 in File LandBaseToken.sol

```
    function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y, bytes calldata data) external {
        require(from != address(0), "from is zero address");
        require(to != address(0), "can't send to zero address");
        bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
        if (msg.sender != from && !metaTx) {
            require(
                _superOperators[msg.sender] ||
                _operatorsForAll[from][msg.sender],
                "not authorized to transferQuad"
            );
        }
        _transferQuad(from, to, size, x, y);
        _numNFTPerAddress[from] -= size * size;
        _numNFTPerAddress[to] += size * size;

        _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data);
    }
```

The code meets the specification.

## Formal Verification Request 197

### transferQuad_change

10, Dec 2019

118.79 ms

Line 300-308 in File LandBaseToken.sol

```
/*@CTK transferQuad_change
  @tag assume_completion
  @pre from != to
  @pre from != address(0)
  @pre to != address(0)
  @pre (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
      _superOperators[msg.sender] \/ _operatorsForAll[from][msg.sender])
  @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - size * size
  @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + size * size
 */
```

Line 309-329 in File LandBaseToken.sol

```
function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
    bytes calldata data) external {
    require(from != address(0), "from is zero address");
    require(to != address(0), "can't send to zero address");
    bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
    if (msg.sender != from && !metaTx) {
        require(
            _superOperators[msg.sender] ||
            _operatorsForAll[from][msg.sender],
            "not authorized to transferQuad"
        );
    }
    _transferQuad(from, to, size, x, y);
    _numNFTPerAddress[from] -= size * size;
    _numNFTPerAddress[to] += size * size;

    _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
        );
}
```

The code meets the specification.

## Formal Verification Request 198

### _checkBatchReceiverAcceptQuad

10, Dec 2019
92.04 ms

Line 331-335 in File LandBaseToken.sol

```
/*@CTK _checkBatchReceiverAcceptQuad
  @tag assume_completion
  @pre size >= 1
  @pre GRID_SIZE == 408
 */
```

Line 336-362 in File LandBaseToken.sol

```
function _checkBatchReceiverAcceptQuad(
    address operator,
```

```
        address from,
        address to,
        uint256 size,
        uint256 x,
        uint256 y,
        bytes memory data
    ) internal {
        if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
            uint256[] memory ids = new uint256[](size*size);
            /*@CTK _checkBatchRecerverAcceptQuad_forloop
              @inv i <= size * size
              @pre size >= 1
              @pre GRID_SIZE == 408
              @post i == size * size
              @post !__should_return
             */
            for (uint256 i = 0; i < size*size; i++) {
                ids[i] = _idInPath(i, size, x, y);
            }
            require(
                _checkOnERC721BatchReceived(operator, from, to, ids, data),
                "erc721 batch transfer rejected by to"
            );
        }
    }
```

The code meets the specification.

## Formal Verification Request 199

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

10, Dec 2019
61.08 ms

Line 371 in File LandBaseToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 380-427 in File LandBaseToken.sol

```
    function batchTransferQuad(
        address from,
        address to,
        uint256[] calldata sizes,
        uint256[] calldata xs,
        uint256[] calldata ys,
        bytes calldata data
    ) external {
        require(from != address(0), "from is zero address");
        require(to != address(0), "can't send to zero address");
        require(sizes.length == xs.length && xs.length == ys.length, "invalid data");
        bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
        if (msg.sender != from && !metaTx) {
            require(
                _superOperators[msg.sender] ||
                _operatorsForAll[from][msg.sender],
                "not authorized to transferMultiQuads"
```

```
        );
    }
    uint256 numTokensTransfered = 0;
    for (uint256 i = 0; i < sizes.length; i++) {
        uint256 size = sizes[i];
        _transferQuad(from, to, size, xs[i], ys[i]);
        numTokensTransfered += size * size;
    }
    _numNFTPerAddress[from] -= numTokensTransfered;
    _numNFTPerAddress[to] += numTokensTransfered;

    if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
        uint256[] memory ids = new uint256[](numTokensTransfered);
        uint256 counter = 0;
        for (uint256 j = 0; j < sizes.length; j++) {
            uint256 size = sizes[j];
            for (uint256 i = 0; i < size*size; i++) {
                ids[counter] = _idInPath(i, size, xs[j], ys[j]);
                counter++;
            }
        }
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}
```

The code meets the specification.

## Formal Verification Request 200

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

10, Dec 2019
19.55 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] ||
```

```
            _operatorsForAll[from][msg.sender],
            "not authorized to transferMultiQuads"
        );
    }
    uint256 numTokensTransfered = 0;
    for (uint256 i = 0; i < sizes.length; i++) {
        uint256 size = sizes[i];
        _transferQuad(from, to, size, xs[i], ys[i]);
        numTokensTransfered += size * size;
    }
    _numNFTPerAddress[from] -= numTokensTransfered;
    _numNFTPerAddress[to] += numTokensTransfered;

    if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
        uint256[] memory ids = new uint256[](numTokensTransfered);
        uint256 counter = 0;
        for (uint256 j = 0; j < sizes.length; j++) {
            uint256 size = sizes[j];
            for (uint256 i = 0; i < size*size; i++) {
                ids[counter] = _idInPath(i, size, xs[j], ys[j]);
                counter++;
            }
        }
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}
```

The code meets the specification.

## Formal Verification Request 201

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

10, Dec 2019
17.73 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) {
```

```
        require(
            _superOperators[msg.sender] ||
            _operatorsForAll[from][msg.sender],
            "not authorized to transferMultiQuads"
        );
    }
    uint256 numTokensTransfered = 0;
    for (uint256 i = 0; i < sizes.length; i++) {
        uint256 size = sizes[i];
        _transferQuad(from, to, size, xs[i], ys[i]);
        numTokensTransfered += size * size;
    }
    _numNFTPerAddress[from] -= numTokensTransfered;
    _numNFTPerAddress[to] += numTokensTransfered;

    if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
        uint256[] memory ids = new uint256[](numTokensTransfered);
        uint256 counter = 0;
        for (uint256 j = 0; j < sizes.length; j++) {
            uint256 size = sizes[j];
            for (uint256 i = 0; i < size*size; i++) {
                ids[counter] = _idInPath(i, size, xs[j], ys[j]);
                counter++;
            }
        }
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}
```

✅ The code meets the specification.

## Formal Verification Request 202

### transferQuad_require

📅 10, Dec 2019
⏱ 13.04 ms

Line 374-379 in File LandBaseToken.sol

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

Line 380-427 in File LandBaseToken.sol

```
function batchTransferQuad(
    address from,
    address to,
    uint256[] calldata sizes,
    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 numTokensTransfered = 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             numTokensTransfered += size * size;
404         }
405         _numNFTPerAddress[from] -= numTokensTransfered;
406         _numNFTPerAddress[to] += numTokensTransfered;
407
408         if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
409             uint256[] memory ids = new uint256[](numTokensTransfered);
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 203

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

📅 10, Dec 2019
⏱ 99.87 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)
            internal {
443         if (size == 1) {
```

```
        uint256 id1x1 = x + y * GRID_SIZE;
        address owner = _ownerOf(id1x1);
        require(owner != address(0), "token does not exist");
        require(owner == from, "not owner in _transferQuad");
        _owners[id1x1] = uint256(to);
    } else {
        _regroup(from, to, size, x, y);
    }
    /*@CTK _transferQuad_loop
      @inv i <= size * size
      @post i == size * size
     */
    for (uint256 i = 0; i < size*size; i++) {
        emit Transfer(from, to, _idInPath(i, size, x, y));
    }
}
```

The code meets the specification.

## Formal Verification Request 204

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

10, Dec 2019
1.57 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)
        internal {
443     if (size == 1) {
444         uint256 id1x1 = x + y * GRID_SIZE;
445         address owner = _ownerOf(id1x1);
446         require(owner != address(0), "token does not exist");
447         require(owner == from, "not owner in _transferQuad");
448         _owners[id1x1] = 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 205

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

10, Dec 2019
24.39 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 206

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

10, Dec 2019
0.49 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 207

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

10, Dec 2019
0.48 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 208

__checkAndClear__require

10, Dec 2019
2.18 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 209

__checkAndClear__change

10, Dec 2019
2.56 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
```

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

Line 477-485 in File LandBaseToken.sol

```
   function _checkAndClear(address from, uint256 id) internal returns(bool) {
       uint256 owner = _owners[id];
       if (owner != 0) {
           require(address(owner) == from, "not owner");
           _owners[id] = 0;
           return true;
       }
       return false;
   }
```

✅ The code meets the specification.

## Formal Verification Request 210

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

📅 10, Dec 2019
⏱ 67.02 ms

Line 487 in File LandBaseToken.sol

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

Line 496-519 in File LandBaseToken.sol

```
   function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
        {
       require(x % size == 0 && y % size == 0, "Invalid coordinates");
       require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");

       if (size == 3) {
           _regroup3x3(from, to, x, y, true);
       } else if (size == 6) {
           _regroup6x6(from, to, x, y, true);
       } else if (size == 12) {
           _regroup12x12(from, to, x, y, true);
       } else if (size == 24) {
           _regroup24x24(from, to, x, y, true);
       } else {
           require(false, "Invalid size");
       }
   }
```

❌ This code violates the specification.

```
Counter Example:
Before Execution:
   Input = {
       from = 0
       size = 3
       to = 0
       x = 18
       y = 153
```

```
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 0,
            "LAYER": 0,
            "LAYER_1x1": 0,
            "LAYER_3x3": 0,
            "LAYER_6x6": 0,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "AAAA",
            "ERC165ID": "AAAA",
            "ERC721_MANDATORY_RECEIVER": "CCCC",
            "_numNFTPerAddress": [
              {
                "key": 128,
                "value": 64
              },
              {
                "key": 0,
                "value": 32
              },
              {
                "key": 136,
                "value": 1
              },
              {
```

```
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_owners": [
          {
            "key": 32,
            "value": 64
          },
          {
            "key": 96,
            "value": 16
          },
          {
            "key": 64,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 4,
            "value": 16
          },
          {
            "key": 64,
            "value": 2
          },
          {
            "key": 32,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 32
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 0,
            "value": false
          },
          {
            "key": "ALL_OTHERS",
```

```
              "value": true
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ]
        }
      }
    }
  ]

After Execution:
  Input = {
    from = 0
    size = 3
    to = 0
    x = 18
    y = 153
  }
  This = 0
  Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = true
    __has_returned = false
    __reverted = false
    msg = {
      "gas": 0,
      "sender": 0,
      "value": 0
    }
  }
  Other = {
    block = {
      "number": 0,
      "timestamp": 0
    }
  }
  Address_Map = [
    {
      "key": "ALL_OTHERS",
      "value": {
        "contract_name": "LandBaseToken",
        "balance": 0,
        "contract": {
          "GRID_SIZE": 0,
          "LAYER": 0,
          "LAYER_1x1": 0,
          "LAYER_3x3": 0,
          "LAYER_6x6": 0,
          "LAYER_12x12": 0,
          "LAYER_24x24": 0,
          "_minters": [
            {
```

```
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_ERC721_RECEIVED": "AAAA",
        "_ERC721_BATCH_RECEIVED": "AAAA",
        "ERC165ID": "AAAA",
        "ERC721_MANDATORY_RECEIVER": "CCCC",
        "_numNFTPerAddress": [
          {
            "key": 128,
            "value": 64
          },
          {
            "key": 0,
            "value": 32
          },
          {
            "key": 136,
            "value": 1
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_owners": [
          {
            "key": 32,
            "value": 64
          },
          {
            "key": 96,
            "value": 16
          },
          {
            "key": 64,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 4,
```

```
              "value": 16
            },
            {
              "key": 64,
              "value": 2
            },
            {
              "key": 32,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 32
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": 0,
              "value": false
            },
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ]
        }
      }
    }
  ]
```


## Formal Verification Request 211

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

📅 10, Dec 2019
⏱ 4.58 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       require(x % size == 0 && y % size == 0, "Invalid coordinates");
498       require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
499
500       if (size == 3) {
501           _regroup3x3(from, to, x, y, true);
502       } else if (size == 6) {
```

```
            _regroup6x6(from, to, x, y, true);
        } else if (size == 12) {
            _regroup12x12(from, to, x, y, true);
        } else if (size == 24) {
            _regroup24x24(from, to, x, y, true);
        } else {
            require(false, "Invalid size");
        }
    }
```

The code meets the specification.

## Formal Verification Request 212

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

10, Dec 2019
4.19 ms

Line 489 in File LandBaseToken.sol

```
    //@CTK NO_ASF
```

Line 496-519 in File LandBaseToken.sol

```
    function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
        {
        require(x % size == 0 && y % size == 0, "Invalid coordinates");
        require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");

        if (size == 3) {
            _regroup3x3(from, to, x, y, true);
        } else if (size == 6) {
            _regroup6x6(from, to, x, y, true);
        } else if (size == 12) {
            _regroup12x12(from, to, x, y, true);
        } else if (size == 24) {
            _regroup24x24(from, to, x, y, true);
        } else {
            require(false, "Invalid size");
        }
    }
```

The code meets the specification.

## Formal Verification Request 213

**__regroup__require**

10, Dec 2019
7.68 ms

Line 490-495 in File LandBaseToken.sol

```
    /*@CTK _regroup_require
      @tag assume_completion
```

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

Line 496-519 in File LandBaseToken.sol

```
  function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
      {
     require(x % size == 0 && y % size == 0, "Invalid coordinates");
     require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");

     if (size == 3) {
        _regroup3x3(from, to, x, y, true);
     } else if (size == 6) {
        _regroup6x6(from, to, x, y, true);
     } else if (size == 12) {
        _regroup12x12(from, to, x, y, true);
     } else if (size == 24) {
        _regroup24x24(from, to, x, y, true);
     } else {
        require(false, "Invalid size");
     }
  }
```

The code meets the specification.

## Formal Verification Request 214

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

10, Dec 2019
2108.97 ms

Line 521 in File LandBaseToken.sol

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

Line 529-554 in File LandBaseToken.sol

```
  function _regroup3x3(address from, address to, uint256 x, uint256 y, bool set) internal
      returns (bool) {
     uint256 id = x + y * GRID_SIZE;
     uint256 quadId = LAYER_3x3 + id;
     bool ownerOfAll = true;
     for (uint256 xi = x; xi < x+3; xi++) {
        for (uint256 yi = y; yi < y+3; yi++) {
           ownerOfAll = _checkAndClear(from, xi + yi * GRID_SIZE) && ownerOfAll;
        }
     }
     if(set) {
        if(!ownerOfAll) {
           require(
              _owners[quadId] == 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),
                "not owner of all sub quads nor parent quads"
            );
        }
        _owners[quadId] = uint256(to);
        return true;
    }
    return ownerOfAll;
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 0
        set = true
        to = 0
        x = 192
        y = 27
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 38,
            "LAYER": 0,
            "LAYER_1x1": 0,
            "LAYER_3x3": 62,
            "LAYER_6x6": 0,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": "ALL_OTHERS",
```

```
            "value": false
          }
        ],
        "_ERC721_RECEIVED": "\u007f\u007f\u007f\u007f",
        "_ERC721_BATCH_RECEIVED": "\u007f\u007f\u007f\u007f",
        "ERC165ID": "\u007f\u007f\u007f\u007f",
        "ERC721_MANDATORY_RECEIVER": "\u007f\u007f\u007f\u007f",
        "_numNFTPerAddress": [
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 64,
            "value": 8
          },
          {
            "key": 128,
            "value": 32
          },
          {
            "key": 4,
            "value": 16
          },
          {
            "key": 8,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 62
          }
        ],
        "_owners": [
          {
            "key": 0,
            "value": 1
          },
          {
            "key": 128,
            "value": 32
          },
          {
            "key": 64,
            "value": 8
          },
          {
            "key": 8,
            "value": 0
          },
          {
            "key": 4,
            "value": 16
          },
          {
            "key": "ALL_OTHERS",
            "value": 62
          }
```

```
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 0,
            "value": 64
          },
          {
            "key": 64,
            "value": 64
          },
          {
            "key": 4,
            "value": 32
          },
          {
            "key": 8,
            "value": 1
          },
          {
            "key": "ALL_OTHERS",
            "value": 62
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 0,
            "value": true
          },
          {
            "key": 16,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": 32,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
```

```
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
    from = 0
    set = true
    to = 0
    x = 192
    y = 27
  }
  This = 0
  Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = true
    __has_returned = true
    __reverted = false
    msg = {
      "gas": 0,
      "sender": 0,
      "value": 0
    }
  }
  Other = {
    __return = true
    block = {
      "number": 0,
      "timestamp": 0
    }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "LandBaseToken",
        "balance": 0,
        "contract": {
          "GRID_SIZE": 38,
          "LAYER": 0,
          "LAYER_1x1": 0,
          "LAYER_3x3": 62,
          "LAYER_6x6": 0,
          "LAYER_12x12": 0,
          "LAYER_24x24": 0,
          "_minters": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
```

```
        "_ERC721_RECEIVED": "\u007f\u007f\u007f\u007f",
        "_ERC721_BATCH_RECEIVED": "\u007f\u007f\u007f\u007f",
        "ERC165ID": "\u007f\u007f\u007f\u007f",
        "ERC721_MANDATORY_RECEIVER": "\u007f\u007f\u007f\u007f",
        "_numNFTPerAddress": [
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 64,
            "value": 8
          },
          {
            "key": 128,
            "value": 32
          },
          {
            "key": 4,
            "value": 16
          },
          {
            "key": 8,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 62
          }
        ],
        "_owners": [
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 64,
            "value": 8
          },
          {
            "key": 128,
            "value": 32
          },
          {
            "key": 4,
            "value": 16
          },
          {
            "key": 8,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 62
          }
        ],
        "_operatorsForAll": [
          {
```

```
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 0,
            "value": 64
          },
          {
            "key": 64,
            "value": 64
          },
          {
            "key": 4,
            "value": 32
          },
          {
            "key": 8,
            "value": 1
          },
          {
            "key": "ALL_OTHERS",
            "value": 62
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 0,
            "value": true
          },
          {
            "key": 16,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": 32,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      }
    }
```

```
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

## Formal Verification Request 215

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

10, Dec 2019
2.11 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                   _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),
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   }
```

The code meets the specification.

## Formal Verification Request 216

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

10, Dec 2019
1.2 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),
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    }
```

✅ The code meets the specification.

## Formal Verification Request 217

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

📅 10, Dec 2019
⏱ 2008.03 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);
```

```
            uint256 id3x3 = LAYER_3x3 + xi + yi * GRID_SIZE;
            uint256 owner3x3 = _owners[id3x3];
            if (owner3x3 != 0) {
                if(!ownAllIndividual) {
                    require(owner3x3 == uint256(from), "not owner of 3x3 quad");
                }
                _owners[id3x3] = 0;
            }
            ownerOfAll = (ownAllIndividual || owner3x3 != 0) && ownerOfAll;
        }
    }
    if(set) {
        if(!ownerOfAll) {
            require(
                _owners[quadId] == 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),
                "not owner of all sub quads nor parent quads"
            );
        }
        _owners[quadId] = uint256(to);
        return true;
    }
    return ownerOfAll;
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 0
        set = false
        to = 0
        x = 33
        y = 163
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
```

```
Address_Map = [
  {
    "key": 0,
    "value": {
      "contract_name": "LandBaseToken",
      "balance": 0,
      "contract": {
        "GRID_SIZE": 183,
        "LAYER": 0,
        "LAYER_1x1": 0,
        "LAYER_3x3": 0,
        "LAYER_6x6": 140,
        "LAYER_12x12": 0,
        "LAYER_24x24": 0,
        "_minters": [
          {
            "key": 64,
            "value": true
          },
          {
            "key": 0,
            "value": true
          },
          {
            "key": 8,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
        "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
        "ERC165ID": "\u0081\u0081\u0081\u0081",
        "ERC721_MANDATORY_RECEIVER": "\u0081\u0081\u0081\u0081",
        "_numNFTPerAddress": [
          {
            "key": 68,
            "value": 0
          },
          {
            "key": 0,
            "value": 8
          },
          {
            "key": 4,
            "value": 2
          },
          {
            "key": "ALL_OTHERS",
            "value": 64
          }
        ],
        "_owners": [
          {
            "key": 0,
            "value": 0
```

```
            },
            {
              "key": 32,
              "value": 128
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
          "_operatorsForAll": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": 1,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 32,
```

```
              "value": 128
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": 2,
              "value": true
            },
            {
              "key": 4,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 1,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
    Input = {
        from = 0
        set = false
        to = 0
        x = 33
        y = 163
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
```

```
      __has_returned = true
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
    }
    Other = {
      __return = true
      block = {
        "number": 0,
        "timestamp": 0
      }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 183,
            "LAYER": 0,
            "LAYER_1x1": 0,
            "LAYER_3x3": 0,
            "LAYER_6x6": 140,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": 64,
                "value": true
              },
              {
                "key": 0,
                "value": true
              },
              {
                "key": 8,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
            "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
            "ERC165ID": "\u0081\u0081\u0081\u0081",
            "ERC721_MANDATORY_RECEIVER": "\u0081\u0081\u0081\u0081",
            "_numNFTPerAddress": [
              {
                "key": 68,
                "value": 0
              },
              {
                "key": 0,
```

```
            "value": 8
          },
          {
            "key": 4,
            "value": 2
          },
          {
            "key": "ALL_OTHERS",
            "value": 64
          }
        ],
        "_owners": [
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 32,
            "value": 128
          },
          {
            "key": 64,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 64
          }
        ],
        "_operatorsForAll": [
          {
            "key": 0,
            "value": [
              {
                "key": 0,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          },
          {
            "key": 1,
            "value": [
              {
                "key": 0,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          },
          {
            "key": "ALL_OTHERS",
```

```
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 32,
              "value": 128
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": 2,
              "value": true
            },
            {
              "key": 4,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": 1,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
```

```
]
```

## Formal Verification Request 218

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

10, Dec 2019
2.27 ms

Line 557 in File LandBaseToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 559-592 in File LandBaseToken.sol

```
function _regroup6x6(address from, address to, uint256 x, uint256 y, bool set) internal returns (bool) {
    uint256 id = x + y * GRID_SIZE;
    uint256 quadId = LAYER_6x6 + id;
    bool ownerOfAll = true;
    for (uint256 xi = x; xi < x+6; xi += 3) {
        for (uint256 yi = y; yi < y+6; yi += 3) {
            bool ownAllIndividual = _regroup3x3(from, to, xi, yi, false);
            uint256 id3x3 = LAYER_3x3 + xi + yi * GRID_SIZE;
            uint256 owner3x3 = _owners[id3x3];
            if (owner3x3 != 0) {
                if(!ownAllIndividual) {
                    require(owner3x3 == uint256(from), "not owner of 3x3 quad");
                }
                _owners[id3x3] = 0;
            }
            ownerOfAll = (ownAllIndividual || owner3x3 != 0) && ownerOfAll;
        }
    }
    if(set) {
        if(!ownerOfAll) {
            require(
                _owners[quadId] == 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),
                "not owner of all sub quads nor parent quads"
            );
        }
        _owners[quadId] = uint256(to);
        return true;
    }
    return ownerOfAll;
}
```

The code meets the specification.

## Formal Verification Request 219

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

10, Dec 2019
1.34 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(
                           from) ||
582                    _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(
                           from),
583                    "not owner of all sub quads nor parent quads"
584                );
585            }
586            _owners[quadId] = uint256(to);
587            return true;
588        }
589        return ownerOfAll;
590    }
```

The code meets the specification.

## Formal Verification Request 220

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

10, Dec 2019
1031.4 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(
                        from),
620                     "not owner of all sub quads nor parent quads"
621                 );
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 = {
```

```
        "gas": 0,
        "sender": 0,
        "value": 0
      }
  }
  Other = {
      __return = false
      block = {
        "number": 0,
        "timestamp": 0
      }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "LandBaseToken",
        "balance": 0,
        "contract": {
          "GRID_SIZE": 17,
          "LAYER": 0,
          "LAYER_1x1": 0,
          "LAYER_3x3": 0,
          "LAYER_6x6": 0,
          "LAYER_12x12": 130,
          "LAYER_24x24": 0,
          "_minters": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
          "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
          "ERC165ID": "\u0081\u0081\u0081\u0081",
          "ERC721_MANDATORY_RECEIVER": "\u00c2\u00c2\u00c2\u00c2",
          "_numNFTPerAddress": [
            {
              "key": 18,
              "value": 32
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 4,
              "value": 2
            },
            {
              "key": 8,
              "value": 8
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
```

```
          "_owners": [
            {
              "key": 2,
              "value": 72
            },
            {
              "key": 0,
              "value": 8
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
          "_operatorsForAll": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": 1,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
```

```
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 64,
            "value": 32
          },
          {
            "key": 8,
            "value": 0
          },
          {
            "key": 1,
            "value": 4
          },
          {
            "key": "ALL_OTHERS",
            "value": 64
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      }
    }
  },
  {
    "key": "ALL_OTHERS",
    "value": "EmptyAddress"
  }
]

After Execution:
  Input = {
    from = 0
    set = false
    to = 0
    x = 8
    y = 113
  }
  This = 0
```

```
Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = true
    __has_returned = true
    __reverted = false
    msg = {
      "gas": 0,
      "sender": 0,
      "value": 0
    }
}
Other = {
    __return = true
    block = {
      "number": 0,
      "timestamp": 0
    }
}
Address_Map = [
  {
    "key": 0,
    "value": {
      "contract_name": "LandBaseToken",
      "balance": 0,
      "contract": {
        "GRID_SIZE": 17,
        "LAYER": 0,
        "LAYER_1x1": 0,
        "LAYER_3x3": 0,
        "LAYER_6x6": 0,
        "LAYER_12x12": 130,
        "LAYER_24x24": 0,
        "_minters": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_ERC721_RECEIVED": "\u0081\u0081\u0081\u0081",
        "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
        "ERC165ID": "\u0081\u0081\u0081\u0081",
        "ERC721_MANDATORY_RECEIVER": "\u00c2\u00c2\u00c2\u00c2",
        "_numNFTPerAddress": [
          {
            "key": 18,
            "value": 32
          },
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 4,
            "value": 2
          },
          {
            "key": 8,
```

```
              "value": 8
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
          "_owners": [
            {
              "key": 2,
              "value": 72
            },
            {
              "key": 0,
              "value": 8
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
          "_operatorsForAll": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": 1,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
```

```
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 64,
              "value": 32
            },
            {
              "key": 8,
              "value": 0
            },
            {
              "key": 1,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 64
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

## Formal Verification Request 221

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

10, Dec 2019
2.06 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)
           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(
                           from),
620                    "not owner of all sub quads nor parent quads"
621                );
622            }
623            _owners[quadId] = uint256(to);
624            return true;
625        }
626        return ownerOfAll;
627    }
```

The code meets the specification.

## Formal Verification Request 222

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

10, Dec 2019
1.03 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(
                    from),
620                 "not owner of all sub quads nor parent quads"
621             );
622         }
623         _owners[quadId] = uint256(to);
624         return true;
625     }
626     return ownerOfAll;
627 }
```

The code meets the specification.

## Formal Verification Request 223

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

10, Dec 2019
3557.81 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;
```

```
        for (uint256 xi = x; xi < x+24; xi += 12) {
            for (uint256 yi = y; yi < y+24; yi += 12) {
                bool ownAllIndividual = _regroup12x12(from, to, xi, yi, false);
                uint256 id12x12 = LAYER_12x12 + xi + yi * GRID_SIZE;
                uint256 owner12x12 = _owners[id12x12];
                if (owner12x12 != 0) {
                    if(!ownAllIndividual) {
                        require(owner12x12 == uint256(from), "not owner of 12x12 quad");
                    }
                    _owners[id12x12] = 0;
                }
                ownerOfAll = (ownAllIndividual || owner12x12 != 0) && ownerOfAll;
            }
        }
        if(set) {
            if(!ownerOfAll) {
                require(
                    _owners[quadId] == uint256(from),
                    "not owner of all sub quads not parent quad"
                );
            }
            _owners[quadId] = uint256(to);
            return true;
        }
        return ownerOfAll || _owners[quadId] == uint256(from);
    }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 0
        set = false
        to = 0
        x = 128
        y = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
```

```
    {
      "key": 0,
      "value": {
        "contract_name": "LandBaseToken",
        "balance": 0,
        "contract": {
          "GRID_SIZE": 0,
          "LAYER": 0,
          "LAYER_1x1": 0,
          "LAYER_3x3": 0,
          "LAYER_6x6": 0,
          "LAYER_12x12": 0,
          "LAYER_24x24": 128,
          "_minters": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_ERC721_RECEIVED": "AAAA",
          "_ERC721_BATCH_RECEIVED": "CCCC",
          "ERC165ID": "CCCC",
          "ERC721_MANDATORY_RECEIVER": "CCCC",
          "_numNFTPerAddress": [
            {
              "key": 64,
              "value": 64
            },
            {
              "key": 8,
              "value": 32
            },
            {
              "key": 1,
              "value": 4
            },
            {
              "key": 9,
              "value": 16
            },
            {
              "key": 0,
              "value": 1
            },
            {
              "key": "ALL_OTHERS",
              "value": 2
            }
          ],
          "_owners": [
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 32,
              "value": 0
            },
```

```
            {
              "key": 16,
              "value": 16
            },
            {
              "key": 0,
              "value": 16
            },
            {
              "key": "ALL_OTHERS",
              "value": 2
            }
          ],
          "_operatorsForAll": [
            {
              "key": 8,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            }
          ],
          "_operators": [
            {
              "key": 4,
              "value": 8
            },
            {
              "key": 32,
              "value": 0
```

```
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": 2,
              "value": 8
            },
            {
              "key": 0,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 2
            }
          ],
          "_metaTransactionContracts": [
            {
              "key": 32,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
    from = 0
    set = false
    to = 0
    x = 128
    y = 0
  }
  This = 0
  Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = true
    __has_returned = true
```

```
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
    }
    Other = {
      __return = true
      block = {
        "number": 0,
        "timestamp": 0
      }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 0,
            "LAYER": 0,
            "LAYER_1x1": 0,
            "LAYER_3x3": 0,
            "LAYER_6x6": 0,
            "LAYER_12x12": 0,
            "LAYER_24x24": 128,
            "_minters": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "CCCC",
            "ERC165ID": "CCCC",
            "ERC721_MANDATORY_RECEIVER": "CCCC",
            "_numNFTPerAddress": [
              {
                "key": 64,
                "value": 64
              },
              {
                "key": 8,
                "value": 32
              },
              {
                "key": 1,
                "value": 4
              },
              {
                "key": 9,
                "value": 16
              },
              {
                "key": 0,
                "value": 1
```

```
            },
            {
              "key": "ALL_OTHERS",
              "value": 2
            }
          ],
          "_owners": [
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 16,
              "value": 16
            },
            {
              "key": 0,
              "value": 16
            },
            {
              "key": "ALL_OTHERS",
              "value": 2
            }
          ],
          "_operatorsForAll": [
            {
              "key": 8,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": true
                },
                {
                  "key": "ALL_OTHERS",
                  "value": false
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
```

```
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 4,
            "value": 8
          },
          {
            "key": 32,
            "value": 0
          },
          {
            "key": 64,
            "value": 0
          },
          {
            "key": 2,
            "value": 8
          },
          {
            "key": 0,
            "value": 4
          },
          {
            "key": "ALL_OTHERS",
            "value": 2
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 32,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "_admin": 0,
        "_superOperators": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      }
    }
  },
  {
    "key": "ALL_OTHERS",
    "value": "EmptyAddress"
  }
]
```

## Formal Verification Request 224

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

10, Dec 2019
1.91 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 225

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

10, Dec 2019
1.31 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 226

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

📅 10, Dec 2019
⏱ 0.88 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 227

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

📅 10, Dec 2019
⏱ 0.78 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 {
```

```
                address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE]);
                if (owner12x12 != address(0)) {
                    return owner12x12;
                } else {
                    return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE]);
                }
            }
        }
    }
}
```

The code meets the specification.

## Formal Verification Request 228

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

10, Dec 2019
0.91 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) * 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     }
```

```
}
```

The code meets the specification.

## Formal Verification Request 229

### _ownerOf

10, Dec 2019
14.61 ms

Line 670-674 in File LandBaseToken.sol

```
/*@CTK FAIL "_ownerOf"
  @pre GRID_SIZE == 408
  @pre (id & LAYER) == 0
  @post (_owners[id] != 0) -> (__return == address(_owners[id]))
 */
```

Line 675-701 in File LandBaseToken.sol

```
function _ownerOf(uint256 id) internal view returns (address) {
    require(id & LAYER == 0, "Invalid token id");
    uint256 x = id % GRID_SIZE;
    uint256 y = id / GRID_SIZE;
    uint256 owner1x1 = _owners[id];

    if (owner1x1 != 0) {
        return address(owner1x1); // cast to zero
    } else {
        address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE]);
        if (owner3x3 != address(0)) {
            return owner3x3;
        } else {
            address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE]);
            if (owner6x6 != address(0)) {
                return owner6x6;
            } else {
                address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE]);
                if (owner12x12 != address(0)) {
                    return owner12x12;
                } else {
                    return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE]);
                }
            }
        }
    }
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
```

```
        id = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "LandBaseToken",
          "balance": 0,
          "contract": {
            "GRID_SIZE": 152,
            "LAYER": 0,
            "LAYER_1x1": 0,
            "LAYER_3x3": 0,
            "LAYER_6x6": 0,
            "LAYER_12x12": 0,
            "LAYER_24x24": 0,
            "_minters": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "_ERC721_RECEIVED": "AAAA",
            "_ERC721_BATCH_RECEIVED": "\u0081\u0081\u0081\u0081",
            "ERC165ID": "AAAA",
            "ERC721_MANDATORY_RECEIVER": "AAAA",
            "_numNFTPerAddress": [
              {
                "key": 4,
                "value": 16
              },
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
            "_owners": [
              {
```

```
            "key": 80,
            "value": 8
          },
          {
            "key": 8,
            "value": 64
          },
          {
            "key": 0,
            "value": 1
          },
          {
            "key": 128,
            "value": 2
          },
          {
            "key": 64,
            "value": 128
          },
          {
            "key": "ALL_OTHERS",
            "value": 32
          }
        ],
        "_operatorsForAll": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": true
              }
            ]
          }
        ],
        "_operators": [
          {
            "key": 80,
            "value": 0
          },
          {
            "key": 2,
            "value": 0
          },
          {
            "key": 0,
            "value": 32
          },
          {
            "key": "ALL_OTHERS",
            "value": 8
          }
        ],
        "_metaTransactionContracts": [
          {
            "key": 0,
            "value": true
          },
```

```
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_admin": 0,
          "_superOperators": [
            {
              "key": "ALL_OTHERS",
              "value": true
            }
          ]
        }
      }
    }
  ]

Function invocation is reverted.
```

## Formal Verification Request 230

### __checkBatchRecerverAcceptQuad__forloop___Generated

📅 10, Dec 2019
⏱ 235.45 ms

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

```
347         /*@CTK _checkBatchRecerverAcceptQuad_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 _checkBatchRecerverAcceptQuad_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 231

### __transferQuad__loop___Generated

📅 10, Dec 2019
⏱ 27.15 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

```
pragma solidity 0.5.9;

import "../sandbox-private-contracts/src/Land.sol";
import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC20.sol";
import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/
    MetaTransactionReceiver.sol";


/**
 * @title Land Sale contract
 * @notice This contract mananges the sale of our lands
 */
contract LandSale is MetaTransactionReceiver {

    uint256 internal constant GRID_SIZE = 408; // 408 is the size of the Land

    Land internal _land;
    ERC20 internal _sand;
    address payable internal _wallet;
    uint256 internal _expiryTime;
    bytes32 internal _merkleRoot;

    event LandQuadPurchased(
        address indexed buyer,
        address indexed to,
        uint256 indexed topCornerId,
        uint256 size,
        uint256 price
    );

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK LandSale
      @tag assume_completion
      @post __post._land == landAddress
      @post __post._sand == sandContractAddress
      @post __post._metaTransactionContracts[initialMetaTx] == true
      @post __post._admin == admin
      @post __post._wallet == initialWalletAddress
      @post __post._merkleRoot == merkleRoot
      @post __post._expiryTime == expiryTime
     */
    constructor(
        address landAddress,
        address sandContractAddress,
        address initialMetaTx,
        address admin,
        address payable initialWalletAddress,
        bytes32 merkleRoot,
        uint256 expiryTime
    ) public {
        _land = Land(landAddress);
        _sand = ERC20(sandContractAddress);
```

```
        _setMetaTransactionProcessor(initialMetaTx, true);
        _admin = admin;
        _wallet = initialWalletAddress;
        _merkleRoot = merkleRoot;
        _expiryTime = expiryTime;
    }

    /// @notice set the wallet receiving the proceeds
    /// @param newWallet address of the new receiving wallet
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setReceivingWallet_require
      @tag assume_completion
      @post newWallet != address(0)
      @post msg.sender == _admin
     */
    /*@CTK setReceivingWallet_change
      @tag assume_completion
      @post __post._wallet == newWallet
     */
    function setReceivingWallet(address payable newWallet) external{
        require(newWallet != address(0), "receiving wallet cannot be zero address");
        require(msg.sender == _admin, "only admin can change the receiving wallet");
        _wallet = newWallet;
    }

    /**
     * @notice buy Land using the merkle proof associated with it
     * @param buyer address that perform the payment
     * @param to address that will own the purchased Land
     * @param reserved the reserved address (if any)
     * @param x x coordinate of the Land
     * @param y y coordinate of the Land
     * @param size size of the pack of Land to purchase
     * @param price amount of Sand to purchase that Land
     * @param proof merkleProof for that particular Land
     * @return The address of the operator
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK buyLandWithSand
      @tag assume_completion
      @pre _expiryTime > block.timestamp
      @post buyer == msg.sender \/ _metaTransactionContracts[msg.sender] == true
      @post reserved == address(0) \/ reserved == buyer
     */
    function buyLandWithSand(
        address buyer,
        address to,
        address reserved,
        uint256 x,
        uint256 y,
        uint256 size,
        uint256 price,
        bytes32 salt,
        bytes32[] calldata proof
```

```
    ) external {
        /* solhint-disable-next-line not-rely-on-time */
        require(block.timestamp < _expiryTime, "sale is over");
        require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "not
            authorized");
        require(reserved == address(0) || reserved == buyer, "cannot buy reserved Land");
        bytes32 leaf = _generateLandHash(x, y, size, price, reserved, salt);

        require(
            _verify(proof, leaf),
            "Invalid land provided"
        );

        require(
            _sand.transferFrom(
                buyer,
                _wallet,
                price
            ),
            "sand transfer failed"
        );

        _land.mintQuad(to, size, x, y, "");
        emit LandQuadPurchased(buyer, to, x + (y * GRID_SIZE), size, price);
    }

    /**
     * @notice Gets the expiry time for the current sale
     * @return The expiry time, as a unix epoch
     */
    /*@CTK getExpiryTime
      @post __return == _expiryTime
     */
    function getExpiryTime() external view returns(uint256) {
        return _expiryTime;
    }

    /**
     * @notice Gets the Merkle root associated with the current sale
     * @return The Merkle root, as a bytes32 hash
     */
    /*@CTK merkleRoot
      @post __return == _merkleRoot
     */
    function merkleRoot() external view returns(bytes32) {
        return _merkleRoot;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    function _generateLandHash(
        uint256 x,
        uint256 y,
        uint256 size,
        uint256 price,
        address reserved,
        bytes32 salt
```

```
    ) internal pure returns (
        bytes32
    ) {
        return keccak256(
            abi.encodePacked(
                x,
                y,
                size,
                price,
                reserved,
                salt
            )
        );
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    function _verify(bytes32[] memory proof, bytes32 leaf) internal view returns (bool) {
        bytes32 computedHash = leaf;

        /*@CTK _verify_loop
          @inv i <= proof.length
          @post i == proof.length
         */
        for (uint256 i = 0; i < proof.length; i++) {
            bytes32 proofElement = proof[i];

            if (computedHash < proofElement) {
                computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
            } else {
                computedHash = keccak256(abi.encodePacked(proofElement, computedHash));
            }
        }

        return computedHash == _merkleRoot;
    }
}
```

File Admin.sol

```
pragma solidity ^0.5.2;

contract Admin {

    address internal _admin;

    event AdminChanged(address oldAdmin, address newAdmin);

    /// @notice gives the current administrator of this contract.
    /// @return the current administrator of this contract.
    /*@CTK getAdmin
      @post __return == _admin
     */
    function getAdmin() external view returns (address) {
        return _admin;
    }

    /// @notice change the administrator to be `newAdmin`.
```

```
    /// @param newAdmin address of the new administrator.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK changeAdmin_requirement
      @tag assume_completion
      @post msg.sender == _admin
     */
    /*@CTK changeAdmin_change
      @tag assume_completion
      @pre msg.sender == _admin
      @post __post._admin == newAdmin
     */
    function changeAdmin(address newAdmin) external {
        require(msg.sender == _admin, "only admin can change admin");
        emit AdminChanged(_admin, newAdmin);
        _admin = newAdmin;
    }

    modifier onlyAdmin() {
        require (msg.sender == _admin, "only admin allowed");
        _;
    }

}
```

File MetaTransactionReceiver.sol

```
pragma solidity ^0.5.2;

import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/Admin.sol";

contract MetaTransactionReceiver is Admin{

    mapping(address => bool) internal _metaTransactionContracts;
    event MetaTransactionProcessor(address metaTransactionProcessor, bool enabled);

    /// @notice Enable or disable the ability of `metaTransactionProcessor` to perform meta-
        tx (metaTransactionProcessor rights).
    /// @param metaTransactionProcessor address that will be given/removed
        metaTransactionProcessor rights.
    /// @param enabled set whether the metaTransactionProcessor is enabled or disabled.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setMetaTransactionProcessor
      @tag assume_completion
      @post msg.sender == _admin
     */
    /*@CTK setMetaTransactionProcessor
      @tag assume_completion
      @inv msg.sender == _admin
      @post __post._metaTransactionContracts[metaTransactionProcessor] == enabled
     */
    function setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled)
        public {
        require(
            msg.sender == _admin,
            "only admin can setup metaTransactionProcessors"
```

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

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK _setMetaTransactionProcessor
      @tag assume_completion
      @post __post._metaTransactionContracts[metaTransactionProcessor] == enabled
     */
    function _setMetaTransactionProcessor(address metaTransactionProcessor, bool enabled) internal {
        _metaTransactionContracts[metaTransactionProcessor] = enabled;
        emit MetaTransactionProcessor(metaTransactionProcessor, enabled);
    }

    /// @notice check whether address `who` is given meta-transaction execution rights.
    /// @param who The address to query.
    /// @return whether the address has meta-transaction execution rights.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK isMetaTransactionProcessor
      @tag assume_completion
      @post __return == _metaTransactionContracts[who]
     */
    function isMetaTransactionProcessor(address who) external view returns(bool) {
        return _metaTransactionContracts[who];
    }
}
```

File ERC721BaseToken.sol

```
/* solhint-disable func-order, code-complexity */
pragma solidity 0.5.9;

import "../sandbox-private-contracts/contracts_common/src/Libraries/AddressUtils.sol";
import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC721TokenReceiver.sol";
import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC721Events.sol";
import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/SuperOperators.sol";
import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/MetaTransactionReceiver.sol";
import "../sandbox-private-contracts/contracts_common/src/Interfaces/ERC721MandatoryTokenReceiver.sol";

contract ERC721BaseToken is ERC721Events, SuperOperators, MetaTransactionReceiver {
    using AddressUtils for address;

    bytes4 internal constant _ERC721_RECEIVED = 0x150b7a02;
    bytes4 internal constant _ERC721_BATCH_RECEIVED = 0x4b808c46;

    bytes4 internal constant ERC165ID = 0x01ffc9a7;
    bytes4 internal constant ERC721_MANDATORY_RECEIVER = 0x5e8bf644;

    mapping (address => uint256) public _numNFTPerAddress;
    mapping (uint256 => uint256) public _owners;
```

```
    mapping (address => mapping(address => bool)) public _operatorsForAll;
    mapping (uint256 => address) public _operators;

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK ERC721BaseToken
      @tag assume_completion
      @post __post._admin == admin
      @post __post._metaTransactionContracts[metaTransactionContract] == true
     */
    constructor(
        address metaTransactionContract,
        address admin
    ) internal {
        _admin = admin;
        _setMetaTransactionProcessor(metaTransactionContract, true);
    }

    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK _transferFrom
      @tag assume_completion
      @pre from != to
      @pre _numNFTPerAddress[from] > 0
      @pre address(_owners[id]) == from
      @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
      @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
      @post __post._owners[id] == uint256(to)
     */
    function _transferFrom(address from, address to, uint256 id) internal {
        _numNFTPerAddress[from]--;
        _numNFTPerAddress[to]++;
        _owners[id] = uint256(to);
        emit Transfer(from, to, id);
    }

    /**
     * @notice Return the number of Land owned by an address
     * @param owner The address to look for
     * @return The number of Land token owned by the address
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK balanceOf_require
      @tag assume_completion
      @post owner != address(0)
     */
    /*@CTK balanceOf_change
      @tag assume_completion
      @pre owner != address(0)
      @post __return == _numNFTPerAddress[owner]
     */
    function balanceOf(address owner) external view returns (uint256) {
        require(owner != address(0), "owner is zero address");
        return _numNFTPerAddress[owner];
```

```
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK _ownerOf
      @post __return == address(_owners[id])
     */
    function _ownerOf(uint256 id) internal view returns (address) {
        return address(_owners[id]);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK _ownerAndOperatorEnabledOf
      @post owner == address(_owners[id])
      @post operatorEnabled == ((_owners[id] / 2**255) == 1)
     */
    function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner,
        bool operatorEnabled) {
        uint256 data = _owners[id];
        owner = address(data);
        operatorEnabled = (data / 2**255) == 1;
    }

    /**
     * @notice Return the owner of a Land
     * @param id The id of the Land
     * @return The address of the owner
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK ownerOf
      @tag assume_completion
      @post owner == address(_owners[id])
      @post owner != address(0)
     */
    function ownerOf(uint256 id) external view returns (address owner) {
        owner = _ownerOf(id);
        require(owner != address(0), "token does not exist");
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK _approveFor
      @post (operator == address(0)) -> (__post._owners[id] == uint256(owner))
      @post (operator != address(0)) -> (__post._owners[id] == uint256(owner) + 2**255)
      @post (operator != address(0)) -> (__post._operators[id] == operator)
     */
    function _approveFor(address owner, address operator, uint256 id) internal {
        if(operator == address(0)) {
            _owners[id] = uint256(owner); // no need to resset the operator, it will be
                overriden next time
        } else {
            _owners[id] = uint256(owner) + 2**255;
            _operators[id] = operator;
```

```
        }
        emit Approval(owner, operator, id);
    }

    /**
     * @notice Approve an operator to spend tokens on the sender behalf
     * @param sender The address giving the approval
     * @param operator The address receiving the approval
     * @param id The id of the token
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK approveFor_require
      @tag assume_completion
      @post sender != address(0)
      @post sender == address(_owners[id])
      @post (msg.sender == sender) || (_metaTransactionContracts[msg.sender]) || (
          _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])
     */
    /*@CTK approveFor_change
      @tag assume_completion
      @pre sender != address(0)
      @pre sender == address(_owners[id])
      @pre (msg.sender == sender) || (_metaTransactionContracts[msg.sender]) || (
          _superOperators[msg.sender]) || (_operatorsForAll[sender][msg.sender])
      @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
      @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
          2**255)
      @post (operator != address(0)) -> (__post._operators[id] == operator)
     */
    function approveFor(
        address sender,
        address operator,
        uint256 id
    ) external {
        address owner = _ownerOf(id);
        require(sender != address(0), "sender is zero address");
        require(
            msg.sender == sender ||
            _metaTransactionContracts[msg.sender] ||
            _superOperators[msg.sender] ||
            _operatorsForAll[sender][msg.sender],
            "not authorized to approve"
        );
        require(owner == sender, "owner != sender");
        _approveFor(owner, operator, id);
    }

    /**
     * @notice Approve an operator to spend tokens on the sender behalf
     * @param operator The address receiving the approval
     * @param id The id of the token
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK approve_require
```

```
     @tag assume_completion
     @post address(_owners[id]) != address(0)
     @post (msg.sender == address(_owners[id])) || (_superOperators[msg.sender]) || (
         _operatorsForAll[address(_owners[id])][msg.sender])
    */
    /*@CTK approve_change
     @tag assume_completion
     @pre address(_owners[id]) != address(0)
     @pre (msg.sender == address(_owners[id])) || (_superOperators[msg.sender]) || (
         _operatorsForAll[address(_owners[id])][msg.sender])
     @post (operator == address(0)) -> (__post._owners[id] == uint256(_owners[id]))
     @post (operator != address(0)) -> (__post._owners[id] == uint256(_owners[id]) +
         2**255)
     @post (operator != address(0)) -> (__post._operators[id] == operator)
    */
    function approve(address operator, uint256 id) external {
        address owner = _ownerOf(id);
        require(owner != address(0), "token does not exist");
        require(
            owner == msg.sender ||
            _superOperators[msg.sender] ||
            _operatorsForAll[owner][msg.sender],
            "not authorized to approve"
        );
        _approveFor(owner, operator, id);
    }

    /**
     * @notice Get the approved operator for a specific token
     * @param id The id of the token
     * @return The address of the operator
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK getApproved
     @tag assume_completion
     @post address(_owners[id]) != address(0)
     @post ((_owners[id] / 2**255) == 1) -> (__return == _operators[id])
     @post ((_owners[id] / 2**255) != 1) -> (__return == address(0))
    */
    function getApproved(uint256 id) external view returns (address) {
        (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
        require(owner != address(0), "token does not exist");
        if (operatorEnabled) {
            return _operators[id];
        } else {
            return address(0);
        }
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK _checkTransfer
     @tag assume_completion
     @post address(_owners[id]) != address(0)
     @post from == _owners[id]
```

```
    @post to != address(0)
    @post (msg.sender != from) && (_metaTransactionContracts[msg.sender] == false) ->
        _superOperators[msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id
        ] / 2**255) == 1) && _operators[id] == msg.sender)
   */
  function _checkTransfer(address from, address to, uint256 id) internal view returns (
      bool isMetaTx) {
      (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
      require(owner != address(0), "token does not exist");
      require(owner == from, "not owner in _checkTransfer");
      require(to != address(0), "can't send to zero address");
      isMetaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
      if (msg.sender != from && !isMetaTx) {
          require(
              _superOperators[msg.sender] ||
              _operatorsForAll[from][msg.sender] ||
              (operatorEnabled && _operators[id] == msg.sender),
              "not approved to transfer"
          );
      }
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK _checkInterfaceWith10000Gas
    @tag assume_completion
    @post __return == true
   */
  function _checkInterfaceWith10000Gas(address _contract, bytes4 interfaceId)
      internal
      view
      returns (bool)
  {
      bool success;
      bool result;
      bytes memory call_data = abi.encodeWithSelector(
          ERC165ID,
          interfaceId
      );
      // solium-disable-next-line security/no-inline-assembly
      /*@CTK _checkInterfaceWith10000Gas_assembly
        @tag assume_completion
        @var bool success
        @var bool result
        @post result == true
        @post success == true
       */
      // solium-disable-next-line security/no-inline-assembly
      assembly {
          let call_ptr := add(0x20, call_data)
          let call_size := mload(call_data)
          let output := mload(0x40) // Find empty storage location using "free memory
              pointer"
          mstore(output, 0x0)
          success := staticcall(
              10000,
              _contract,
```

```
            call_ptr,
            call_size,
            output,
            0x20
        ) // 32 bytes
        result := mload(output)
    }
    // (10000 / 63) "not enough for supportsInterface(...)" // consume all gas, so caller can potentially know that there was not enough gas
    assert(gasleft() > 158);
    return success && result;
}

/**
 * @notice Transfer a token between 2 addresses
 * @param from The sender of the token
 * @param to The recipient of the token
 * @param id The id of the token
*/
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK FAIL NO_ASF
/*@CTK transferFrom
  @tag assume_completion
  @pre (from == _owners[id]) && (from != address(0))
  @pre to != 0
  @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || _superOperators[msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id] / 2**255) == 1) && _operators[id] == msg.sender)
  @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
  @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
  @post __post._owners[id] == uint256(to)
 */
function transferFrom(address from, address to, uint256 id) external {
    bool metaTx = _checkTransfer(from, to, id);
    _transferFrom(from, to, id);
    if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
        require(
            _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, ""),
            "erc721 transfer rejected by to"
        );
    }
}

/**
 * @notice Transfer a token between 2 addresses letting the receiver knows of the transfer
 * @param from The sender of the token
 * @param to The recipient of the token
 * @param id The id of the token
 * @param data Additional data
 */
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK FAIL NO_ASF
/*@CTK safeTransferFrom
  @tag assume_completion
  @pre (from == _owners[id]) && (from != address(0))
```

```
     @pre to != address(0)
     @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || _superOperators[msg.sender] || _operatorsForAll[from][msg.sender] || (((_owners[id] / 2**255) == 1) && _operators[id] == msg.sender)
     @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
     @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + 1
     @post __post._owners[id] == uint256(to)
    */
    function safeTransferFrom(address from, address to, uint256 id, bytes memory data) public {
        bool metaTx = _checkTransfer(from, to, id);
        _transferFrom(from, to, id);
        if (to.isContract()) {
            require(
                _checkOnERC721Received(metaTx ? from : msg.sender, from, to, id, data),
                "ERC721: transfer rejected by to"
            );
        }
    }

    /**
     * @notice Transfer a token between 2 addresses letting the receiver knows of the transfer
     * @param from The send of the token
     * @param to The recipient of the token
     * @param id The id of the token
     */
    function safeTransferFrom(address from, address to, uint256 id) external {
        safeTransferFrom(from, to, id, "");
    }

    /**
     * @notice Transfer many tokens between 2 addresses
     * @param from The sender of the token
     * @param to The recipient of the token
     * @param ids The ids of the tokens
     * @param data additional data
     */
    function batchTransferFrom(address from, address to, uint256[] calldata ids, bytes calldata data) external {
        _batchTransferFrom(from, to, ids, data, false);
    }

    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK FAIL "_batchTransferFrom"
      @tag assume_completion
      @pre from != address(0)
      @pre to != address(0)
      @post _numNFTPerAddress[from] + _numNFTPerAddress[to] == (__post._numNFTPerAddress[from] + __post._numNFTPerAddress[to])
     */
    function _batchTransferFrom(address from, address to, uint256[] memory ids, bytes memory data, bool safe) internal {
        bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
        bool authorized = msg.sender == from ||
            metaTx ||
```

```
        _superOperators[msg.sender] ||
        _operatorsForAll[from][msg.sender];

    require(from != address(0), "from is zero address");
    require(to != address(0), "can't send to zero address");

    uint256 numTokens = ids.length;
    /*@CTK "_batchTransferFrom_loop"
      @pre from != address(0)
      @pre to != address(0)
      @pre numTokens < 5
      @inv this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @inv this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @inv ids == ids__pre
      @pre forall j: uint. (j >= 0 /\ j < numTokens) -> (address(_owners[ids[j]]) == from)
      @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))
      @inv i <= numTokens
      @inv forall j: uint. (j >= 0 /\ j < i) -> this._owners[ids[j]] == uint256(to)
      @inv numTokens == numTokens__pre
      @post (this._numNFTPerAddress[from] + this._numNFTPerAddress[to]) == (this__pre._numNFTPerAddress[from] + this__pre._numNFTPerAddress[to])
      @post this._numNFTPerAddress[from] == this__pre._numNFTPerAddress[from]
      @post this._numNFTPerAddress[to] == this__pre._numNFTPerAddress[to]
      @post i == numTokens
      @post !__should_return
     */
    for(uint256 i = 0; i < numTokens; i ++) {
        uint256 id = ids[i];
        (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
        require(owner == from, "not owner in batchTransferFrom");
        require(authorized || (operatorEnabled && _operators[id] == msg.sender), "not authorized");
        _owners[id] = uint256(to);
        // emit Transfer(from, to, id);
    }
    if (from != to) {
        _numNFTPerAddress[from] -= numTokens;
        _numNFTPerAddress[to] += numTokens;
    }
    if (to.isContract() && (safe || _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER))) {
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}

/**
 * @notice Transfer many tokens between 2 addresses ensuring the receiving contract has a receiver method
 * @param from The sender of the token
 * @param to The recipient of the token
 * @param ids The ids of the tokens
```

```
     * @param data additional data
    */
    function safeBatchTransferFrom(address from, address to, uint256[] calldata ids, bytes
        calldata data) external {
        _batchTransferFrom(from, to, ids, data, true);
    }

    /**
     * @notice Check if the contract supports an interface
     * 0x01ffc9a7 is ERC-165
     * 0x80ac58cd is ERC-721
     * @param id The id of the interface
     * @return True if the interface is supported
     */
    /*@CTK supportsInterface
      @tag assume_completion
      @post __return == (id == 0x01ffc9a7) || (id == 0x80ac58cd)
     */
    function supportsInterface(bytes4 id) external pure returns (bool) {
        return id == 0x01ffc9a7 || id == 0x80ac58cd;
    }

    /**
     * @notice Set the approval for an operator to manage all the tokens of the sender
     * @param sender The address giving the approval
     * @param operator The address receiving the approval
     * @param approved The determination of the approval
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK _setApprovalForAll_require
      @tag assume_completion
      @post sender != address(0)
      @post msg.sender == sender \/ _metaTransactionContracts[msg.sender] == true \/
          _superOperators[msg.sender] == true
      @post _superOperators[operator] == false
     */
    /*@CTK _setApprovalForAll_change
      @tag assume_completion
      @pre sender != address(0)
      @pre msg.sender == sender \/ _metaTransactionContracts[msg.sender] == true \/
          _superOperators[msg.sender] == true
      @pre _superOperators[operator] == false
      @post __post._operatorsForAll[sender][operator] == approved
     */
    function setApprovalForAllFor(
        address sender,
        address operator,
        bool approved
    ) external {
        require(sender != address(0), "Invalid sender address");
        require(
            msg.sender == sender ||
            _metaTransactionContracts[msg.sender] ||
            _superOperators[msg.sender],
            "not authorized to approve for all"
        );
```

```
        _setApprovalForAll(sender, operator, approved);
    }

    /**
     * @notice Set the approval for an operator to manage all the tokens of the sender
     * @param operator The address receiving the approval
     * @param approved The determination of the approval
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setApprovalForAll_require
      @tag assume_completion
      @post _superOperators[operator] == false
     */
    /*@CTK setApprovalForAll_change
      @tag assume_completion
      @pre _superOperators[operator] == false
      @post __post._operatorsForAll[msg.sender][operator] == approved
     */
    function setApprovalForAll(address operator, bool approved) external {
        _setApprovalForAll(msg.sender, operator, approved);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK _setApprovalForAll_require
      @tag assume_completion
      @post _superOperators[operator] == false
     */
    /*@CTK _setApprovalForAll_change
      @tag assume_completion
      @pre _superOperators[operator] == false
      @post __post._operatorsForAll[sender][operator] == approved
     */
    function _setApprovalForAll(
        address sender,
        address operator,
        bool approved
    ) internal {
        require(
            !_superOperators[operator],
            "super operator can't have their approvalForAll changed"
        );
        _operatorsForAll[sender][operator] = approved;

        emit ApprovalForAll(sender, operator, approved);
    }

    /**
     * @notice Check if the sender approved the operator
     * @param owner The address of the owner
     * @param operator The address of the operator
     * @return The status of the approval
     */
    //@CTK NO_OVERFLOW
```

```
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK isApprovedForAll
      @post (_operatorsForAll[owner][operator] == true \/ _superOperators[operator] == true)
          -> __return == true
      @post (_operatorsForAll[owner][operator] == false /\ _superOperators[operator] ==
          false) -> __return == false
     */
    function isApprovedForAll(address owner, address operator)
        external
        view
        returns (bool)
    {
        return _operatorsForAll[owner][operator] || _superOperators[operator];
    }

    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK _burn_require
      @tag assume_completion
      @post from == owner
     */
    /*@CTK _burn_change
      @tag assume_completion
      @pre from == owner
      @post __post._owners[id] == 2**160
      @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
     */
    function _burn(address from, address owner, uint256 id) public {
        require(from == owner, "not owner");
        _owners[id] = 2**160; // cannot mint it again
        _numNFTPerAddress[from]--;
        emit Transfer(from, address(0), id);
    }

    /// @notice Burns token `id`.
    /// @param id token which will be burnt.
    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK burn_require
      @tag assume_completion
      @post msg.sender == address(_owners[id])
     */
    /*@CTK burn_change
      @tag assume_completion
      @pre msg.sender == address(_owners[id])
      @post __post._owners[id] == 2**160
      @post __post._numNFTPerAddress[msg.sender] == _numNFTPerAddress[msg.sender] - 1
     */
    function burn(uint256 id) external {
        _burn(msg.sender, _ownerOf(id), id);
    }

    /// @notice Burn token`id` from `from`.
    /// @param from address whose token is to be burnt.
    /// @param id token which will be burnt.
```

```
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK burnFrom_require
      @tag assume_completion
      @post from != address(0)
      @post (msg.sender == from) || _metaTransactionContracts[msg.sender] || ((_owners[id] /
          2**255) == 1 && _operators[id] == msg.sender) || _superOperators[msg.sender] ||
          _operatorsForAll[from][msg.sender]
      @post from == address(_owners[id])
     */
    /*@CTK burnFrom_change
      @tag assume_completion
      @pre from != address(0)
      @pre (msg.sender == from) || _metaTransactionContracts[msg.sender] || ((_owners[id] /
          2**255) == 1 && _operators[id] == msg.sender) || _superOperators[msg.sender] ||
          _operatorsForAll[from][msg.sender]
      @pre from == address(_owners[id])
      @post __post._owners[id] == 2**160
      @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - 1
     */
    function burnFrom(address from, uint256 id) external {
        require(from != address(0), "Invalid sender address");
        (address owner, bool operatorEnabled) = _ownerAndOperatorEnabledOf(id);
        require(
            msg.sender == from ||
            _metaTransactionContracts[msg.sender] ||
            (operatorEnabled && _operators[id] == msg.sender) ||
            _superOperators[msg.sender] ||
            _operatorsForAll[from][msg.sender],
            "not authorized to burn"
        );
        _burn(from, owner, id);
    }

    function _checkOnERC721Received(address operator, address from, address to, uint256
        tokenId, bytes memory _data)
        internal returns (bool)
    {
        bytes4 retval = ERC721TokenReceiver(to).onERC721Received(operator, from, tokenId,
            _data);
        return (retval == _ERC721_RECEIVED);
    }

    function _checkOnERC721BatchReceived(address operator, address from, address to, uint256
        [] memory ids, bytes memory _data)
        internal returns (bool)
    {
        bytes4 retval = ERC721MandatoryTokenReceiver(to).onERC721BatchReceived(operator,
            from, ids, _data);
        return (retval == _ERC721_BATCH_RECEIVED);
    }
}
```

File SuperOperators.sol

```
pragma solidity ^0.5.2;

import "../sandbox-private-contracts/contracts_common/src/BaseWithStorage/Admin.sol";

```

```
contract SuperOperators is Admin {

    mapping(address => bool) internal _superOperators;

    event SuperOperator(address superOperator, bool enabled);

    /// @notice Enable or disable the ability of `superOperator` to transfer tokens of all (
        superOperator rights).
    /// @param superOperator address that will be given/removed superOperator right.
    /// @param enabled set whether the superOperator is enabled or disabled.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setSuperOperator_admin
      @tag assume_completion
      @inv msg.sender == _admin
     */
    /*@CTK setSuperOperator_change
      @tag assume_completion
      @pre msg.sender == _admin
      @post __post._superOperators[superOperator] == enabled
     */
    function setSuperOperator(address superOperator, bool enabled) external {
        require(
            msg.sender == _admin,
            "only admin is allowed to add super operators"
        );
        _superOperators[superOperator] = enabled;
        emit SuperOperator(superOperator, enabled);
    }

    /// @notice check whether address `who` is given superOperator rights.
    /// @param who The address to query.
    /// @return whether the address has superOperator rights.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK isSuperOperator
      @tag assume_completion
      @post __return == _superOperators[who]
     */
    function isSuperOperator(address who) public view returns (bool) {
        return _superOperators[who];
    }
}
```

File AddressUtils.sol

```
pragma solidity ^0.5.2;

library AddressUtils {

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    function toPayable(address _address) internal pure returns (address payable _payable) {
        return address(uint160(_address));
    }

```

```
    function isContract(address addr) internal view returns (bool) {
        // for accounts without code, i.e. `keccak256('')`:
        bytes32 accountHash = 0
            xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;

        bytes32 codehash;
        // solium-disable-next-line security/no-inline-assembly
        assembly {
            codehash := extcodehash(addr)
        }
        return (codehash != 0x0 && codehash != accountHash);
    }
}
```

File Land.sol

```
/* solhint-disable no-empty-blocks */

pragma solidity 0.5.9;

import "../sandbox-private-contracts/src/Land/erc721/LandBaseToken.sol";

contract Land is LandBaseToken {
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK Land
      @tag assume_completion
      @post __post._admin == admin
      @post __post._metaTransactionContracts[metaTransactionContract] == true
     */
    constructor(
        address metaTransactionContract,
        address admin
    ) public LandBaseToken(
        metaTransactionContract,
        admin
    ) {
    }

    /**
     * @notice Return the name of the token contract
     * @return The name of the token contract
     */
    /*@CTK name
      @post __return == "Sandbox's LANDs"
     */
    function name() external pure returns (string memory) {
        return "Sandbox's LANDs";
    }

    /**
     * @notice Return the symbol of the token contract
     * @return The symbol of the token contract
     */
    /*@CTK symbol
      @post __return == "LAND"
     */
    function symbol() external pure returns (string memory) {
```

```
        return "LAND";
    }

    // solium-disable-next-line security/no-assign-params
    //TODO
    function uint2str(uint _i) internal pure returns (string memory) {
        if (_i == 0) {
            return "0";
        }
        uint j = _i;
        uint len;
        while (j != 0) {
            len++;
            j /= 10;
        }
        bytes memory bstr = new bytes(len);
        uint k = len - 1;
        while (_i != 0) {
            bstr[k--] = byte(uint8(48 + _i % 10));
            _i /= 10;
        }
        return string(bstr);
    }

    /**
     * @notice Return the URI of a specific token
     * @param id The id of the token
     * @return The URI of the token
     */
    function tokenURI(uint256 id) public view returns (string memory) {
        require(_ownerOf(id) != address(0), "Id does not exist");
        return
            string(
                abi.encodePacked(
                    "https://api.sandbox.game/lands/",
                    uint2str(id),
                    "/metadata.json"
                )
            );
    }

    /**
     * @notice Check if the contract supports an interface
     * 0x01ffc9a7 is ERC-165
     * 0x80ac58cd is ERC-721
     * 0x5b5e139f is ERC-721 metadata
     * @param id The id of the interface
     * @return True if the interface is supported
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK supportsInterface
      @tag assume_completion
      @post (id == 0x01ffc9a7 \/ id == 0x80ac58cd \/ id == 0x5b5e139f) -> __return == true
      @post (id != 0x01ffc9a7 /\ id != 0x80ac58cd /\ id != 0x5b5e139f) -> __return == false
     */
    function supportsInterface(bytes4 id) external pure returns (bool) {
```

```
        return id == 0x01ffc9a7 || id == 0x80ac58cd || id == 0x5b5e139f;
    }
}
```

File LandBaseToken.sol

```
/* solhint-disable func-order, code-complexity */
pragma solidity 0.5.9;

import "./ERC721BaseToken.sol";

contract LandBaseToken is ERC721BaseToken {
    // Our grid is 408 x 408 lands
    uint256 internal constant GRID_SIZE = 408;

    uint256 internal constant LAYER =    0
        xFF0000000000000000000000000000000000000000000000000000000000000000;
    uint256 internal constant LAYER_1x1 = 0
        x0000000000000000000000000000000000000000000000000000000000000000;
    uint256 internal constant LAYER_3x3 = 0
        x0100000000000000000000000000000000000000000000000000000000000000;
    uint256 internal constant LAYER_6x6 = 0
        x0200000000000000000000000000000000000000000000000000000000000000;
    uint256 internal constant LAYER_12x12 = 0
        x0300000000000000000000000000000000000000000000000000000000000000;
    uint256 internal constant LAYER_24x24 = 0
        x0400000000000000000000000000000000000000000000000000000000000000;

    mapping(address => bool) internal _minters;
    event Minter(address superOperator, bool enabled);

    /// @notice Enable or disable the ability of `minter` to mint tokens
    /// @param minter address that will be given/removed minter right.
    /// @param enabled set whether the minter is enabled or disabled.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setMinter_require
      @tag assume_completion
      @post msg.sender == _admin
     */
    /*@CTK setMinter_change
      @tag assume_completion
      @post __post._minters[minter] == enabled
     */
    function setMinter(address minter, bool enabled) external {
        require(
            msg.sender == _admin,
            "only admin is allowed to add minters"
        );
        _minters[minter] = enabled;
        emit Minter(minter, enabled);
    }

    /// @notice check whether address `who` is given minter rights.
    /// @param who The address to query.
    /// @return whether the address has minter rights.
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
```

```
    //@CTK NO_ASF
    /*@CTK isMinter
      @tag assume_completion
      @post __return == _minters[who]
     */
    function isMinter(address who) public view returns (bool) {
        return _minters[who];
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK LandBaseToken
      @tag assume_completion
      @post __post._admin == admin
      @post __post._metaTransactionContracts[metaTransactionContract] == true
     */
    constructor(
        address metaTransactionContract,
        address admin
    ) public ERC721BaseToken(metaTransactionContract, admin) {
    }

    /// @notice total width of the map
    /// @return width
    /*@CTK width
      @post __return == GRID_SIZE
     */
    function width() external returns(uint256) {
        return GRID_SIZE;
    }

    /// @notice total height of the map
    /// @return height
    /*@CTK height
      @post __return == GRID_SIZE
     */
    function height() external returns(uint256) {
        return GRID_SIZE;
    }

    /// @notice x coordinate of Land token
    /// @param id tokenId
    /// @return the x coordinates
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK x
      @tag assume_completion
      @pre GRID_SIZE == 408
      @pre address(_owners[id]) != address(0)
      @post __return == id % GRID_SIZE
     */
    function x(uint256 id) external returns(uint256) {
        require(_ownerOf(id) != address(0), "token does not exist");
        return id % GRID_SIZE;
    }

```

```
    /// @notice y coordinate of Land token
    /// @param id tokenId
    /// @return the y coordinates
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK y
      @tag assume_completion
      @pre GRID_SIZE == 408
      @pre address(_owners[id]) != address(0)
      @post __return == id / GRID_SIZE
     */
    function y(uint256 id) external returns(uint256) {
        require(_ownerOf(id) != address(0), "token does not exist");
        return id / GRID_SIZE;
    }

    /**
     * @notice Mint a new quad (aligned to a quad tree with size 3, 6, 12 or 24 only)
     * @param to The recipient of the new quad
     * @param size The size of the new quad
     * @param x The top left x coordinate of the new quad
     * @param y The top left y coordinate of the new quad
     * @param data extra data to pass to the transfer
     */
    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK mintQuad_require
      @tag assume_completion
      @pre GRID_SIZE == 408
      @post to != address(0)
      @post _minters[msg.sender] == true
      @post (x % size == 0) /\ (y % size == 0)
      @post (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
      @post (size == 1 \/ size == 3 \/ size == 6 \/ size == 12 \/ size == 24)
      @post _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0
      @post size <= 12 -> _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] ==
          0
      @post size <= 6 -> _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0
      @post size <= 3 -> _owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0
     */
    /*@CTK mintQuad_change
      @tag assume_completion
      @pre GRID_SIZE == 408
      @pre LAYER == 0xFF0000000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_1x1 == 0x0000000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_3x3 == 0x0100000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_6x6 == 0x0200000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_12x12 == 0x0300000000000000000000000000000000000000000000000000000000000000
      @pre LAYER_24x24 == 0x0400000000000000000000000000000000000000000000000000000000000000
      @pre to != address(0)
      @pre _minters[msg.sender] == true
      @pre (x % size == 0) /\ (y % size == 0)
      @pre (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
      @post (size == 1 \/ size == 3 \/ size == 6 \/ size == 12 \/ size == 24)
      @pre _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0
      @pre size <= 12 -> _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0
```

```
    @pre size <= 6 -> _owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0
    @pre size <= 3 -> _owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0
    @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + (size * size)
   */
  function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata data) external {
      require(to != address(0), "to is zero address");
      require(
          isMinter(msg.sender),
          "Only a minter can mint"
      );
      require(x % size == 0 && y % size == 0, "Invalid coordinates");
      require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");

      uint256 quadId;
      uint256 id = x + y * GRID_SIZE;

      if (size == 1) {
          quadId = id;
      } else if (size == 3) {
          quadId = LAYER_3x3 + id;
      } else if (size == 6) {
          quadId = LAYER_6x6 + id;
      } else if (size == 12) {
          quadId = LAYER_12x12 + id;
      } else if (size == 24) {
          quadId = LAYER_24x24 + id;
      } else {
          require(false, "Invalid size");
      }

      require(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == 0, "Already minted as 24x24");

      uint256 toX = x+size;
      uint256 toY = y+size;
      if (size <= 12) {
          require(
              _owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE] == 0,
              "Already minted as 12x12"
          );
      } else {
          /*@*CTK mintQuad_loop1
            @tag assume_completion
            @inv x12i <= x + size
            @post x12i == x + size
           */
          for (uint256 x12i = x; x12i < toX; x12i += 12) {
              for (uint256 y12i = y; y12i < toY; y12i += 12) {
                  uint256 id12x12 = LAYER_12x12 + x12i + y12i * GRID_SIZE;
                  require(_owners[id12x12] == 0, "Already minted as 12x12");
              }
          }
      }

      if (size <= 6) {
          require(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE] == 0, "Already minted as 6x6");
```

```
        } else {
            for (uint256 x6i = x; x6i < toX; x6i += 6) {
                for (uint256 y6i = y; y6i < toY; y6i += 6) {
                    uint256 id6x6 = LAYER_6x6 + x6i + y6i * GRID_SIZE;
                    require(_owners[id6x6] == 0, "Already minted as 6x6");
                }
            }
        }

        if (size <= 3) {
            require(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE] == 0, "Already
                minted as 3x3");
        } else {
            for (uint256 x3i = x; x3i < toX; x3i += 3) {
                for (uint256 y3i = y; y3i < toY; y3i += 3) {
                    uint256 id3x3 = LAYER_3x3 + x3i + y3i * GRID_SIZE;
                    require(_owners[id3x3] == 0, "Already minted as 3x3");
                }
            }
        }

        /*@*CTK mintQuad_loopx
          @tag assume_completion
          @pre GRID_SIZE == 408
          @inv i <= size * size
          @post i == size * size
         */
        for (uint256 i = 0; i < size*size; i++) {
            uint256 id = _idInPath(i, size, x, y);
            require(_owners[id] == 0, "Already minted");
            emit Transfer(address(0), to, id);
        }

        _owners[quadId] = uint256(to);
        _numNFTPerAddress[to] += size * size;

        _checkBatchReceiverAcceptQuad(msg.sender, address(0), to, size, x, y, data);
    }

    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK _idInPath
      @tag assume_completion
      @pre GRID_SIZE == 408
      @post (((i / size) % 2) == 0) -> __return == (x + (i%size)) + ((y + i / size) *
          GRID_SIZE)
      @post (((i / size) % 2) == 1) -> __return == ((x + size) - (1 + i%size)) + ((y + i /
          size) * GRID_SIZE)
     */
    function _idInPath(uint256 i, uint256 size, uint256 x, uint256 y) internal pure returns(
        uint256) {
        uint256 row = i / size;
        if(row % 2 == 0) { // alow ids to follow a path in a quad
            return (x + (i%size)) + ((y + row) * GRID_SIZE);
        } else {
            return ((x + size) - (1 + i%size)) + ((y + row) * GRID_SIZE);
        }
```

```
    }

    /// @notice transfer one quad (aligned to a quad tree with size 3, 6, 12 or 24 only)
    /// @param from current owner of the quad
    /// @param to destination
    /// @param size size of the quad
    /// @param x The top left x coordinate of the quad
    /// @param y The top left y coordinate of the quad
    /// @param data additional data
    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK transferQuad_require
      @tag assume_completion
      @post from != address(0)
      @post to != address(0)
      @post (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
          _superOperators[msg.sender] \/ _operatorsForAll[from][msg.sender])
     */
    /*@CTK transferQuad_change
      @tag assume_completion
      @pre from != to
      @pre from != address(0)
      @pre to != address(0)
      @pre (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
          _superOperators[msg.sender] \/ _operatorsForAll[from][msg.sender])
      @post __post._numNFTPerAddress[from] == _numNFTPerAddress[from] - size * size
      @post __post._numNFTPerAddress[to] == _numNFTPerAddress[to] + size * size
     */
    function transferQuad(address from, address to, uint256 size, uint256 x, uint256 y,
        bytes calldata data) external {
        require(from != address(0), "from is zero address");
        require(to != address(0), "can't send to zero address");
        bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
        if (msg.sender != from && !metaTx) {
            require(
                _superOperators[msg.sender] ||
                _operatorsForAll[from][msg.sender],
                "not authorized to transferQuad"
            );
        }
        _transferQuad(from, to, size, x, y);
        _numNFTPerAddress[from] -= size * size;
        _numNFTPerAddress[to] += size * size;

        _checkBatchReceiverAcceptQuad(metaTx ? from : msg.sender, from, to, size, x, y, data
            );
    }

    /*@CTK _checkBatchReceiverAcceptQuad
      @tag assume_completion
      @pre size >= 1
      @pre GRID_SIZE == 408
     */
    function _checkBatchReceiverAcceptQuad(
        address operator,
        address from,
        address to,
```

```
        uint256 size,
        uint256 x,
        uint256 y,
        bytes memory data
    ) internal {
        if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
            uint256[] memory ids = new uint256[](size*size);
            /*@CTK _checkBatchRecerverAcceptQuad_forloop
              @inv i <= size * size
              @pre size >= 1
              @pre GRID_SIZE == 408
              @post i == size * size
              @post !__should_return
             */
            for (uint256 i = 0; i < size*size; i++) {
                ids[i] = _idInPath(i, size, x, y);
            }
            require(
                _checkOnERC721BatchReceived(operator, from, to, ids, data),
                "erc721 batch transfer rejected by to"
            );
        }
    }

    /// @notice transfer multiple quad (aligned to a quad tree with size 3, 6, 12 or 24 only
        )
    /// @param from current owner of the quad
    /// @param to destination
    /// @param sizes list of sizes for each quad
    /// @param xs list of top left x coordinates for each quad
    /// @param ys list of top left y coordinates for each quad
    /// @param data additional data
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK transferQuad_require
      @tag assume_completion
      @post from != address(0)
      @post to != address(0)
      @post (msg.sender != from /\ _metaTransactionContracts[msg.sender] == false) -> (
          _superOperators[msg.sender] \/ _operatorsForAll[from][msg.sender])
     */
    function batchTransferQuad(
        address from,
        address to,
        uint256[] calldata sizes,
        uint256[] calldata xs,
        uint256[] calldata ys,
        bytes calldata data
    ) external {
        require(from != address(0), "from is zero address");
        require(to != address(0), "can't send to zero address");
        require(sizes.length == xs.length && xs.length == ys.length, "invalid data");
        bool metaTx = msg.sender != from && _metaTransactionContracts[msg.sender];
        if (msg.sender != from && !metaTx) {
            require(
                _superOperators[msg.sender] ||
                _operatorsForAll[from][msg.sender],
```

```
            "not authorized to transferMultiQuads"
        );
    }
    uint256 numTokensTransfered = 0;
    for (uint256 i = 0; i < sizes.length; i++) {
        uint256 size = sizes[i];
        _transferQuad(from, to, size, xs[i], ys[i]);
        numTokensTransfered += size * size;
    }
    _numNFTPerAddress[from] -= numTokensTransfered;
    _numNFTPerAddress[to] += numTokensTransfered;

    if (to.isContract() && _checkInterfaceWith10000Gas(to, ERC721_MANDATORY_RECEIVER)) {
        uint256[] memory ids = new uint256[](numTokensTransfered);
        uint256 counter = 0;
        for (uint256 j = 0; j < sizes.length; j++) {
            uint256 size = sizes[j];
            for (uint256 i = 0; i < size*size; i++) {
                ids[counter] = _idInPath(i, size, xs[j], ys[j]);
                counter++;
            }
        }
        require(
            _checkOnERC721BatchReceived(metaTx ? from : msg.sender, from, to, ids, data),
            "erc721 batch transfer rejected by to"
        );
    }
}

//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@*CTK _transferQuad_require
  @tag assume_completion
  @pre GRID_SIZE == 408
  @post (size == 1) -> (address(_owners[x + y * GRID_SIZE]) != address(0) /\ address(
      _owners[x + y * GRID_SIZE]) == from)
 */
/*@*CTK _transferQuad_change
  @tag assume_completion
  @pre GRID_SIZE == 408
  @pre (size == 1) -> (address(_owners[x + y * GRID_SIZE]) != address(0) /\ address(
      _owners[x + y * GRID_SIZE]) == from)
  @post (size == 1) -> (_owners[x + y * GRID_SIZE] == uint256(to))
 */
function _transferQuad(address from, address to, uint256 size, uint256 x, uint256 y)
    internal {
    if (size == 1) {
        uint256 id1x1 = x + y * GRID_SIZE;
        address owner = _ownerOf(id1x1);
        require(owner != address(0), "token does not exist");
        require(owner == from, "not owner in _transferQuad");
        _owners[id1x1] = uint256(to);
    } else {
        _regroup(from, to, size, x, y);
    }
    /*@CTK _transferQuad_loop
      @inv i <= size * size
      @post i == size * size
```

```
     */
    for (uint256 i = 0; i < size*size; i++) {
        emit Transfer(from, to, _idInPath(i, size, x, y));
    }
}

//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK _checkAndClear_require
  @tag assume_completion
  @post (_owners[id] != 0) -> (address(_owners[id]) == from)
 */
/*@CTK _checkAndClear_change
  @tag assume_completion
  @pre (_owners[id] != 0) -> (address(_owners[id]) == from)
  @post _owners[id] == 0 -> __return == false
  @post (_owners[id] != 0) -> __post._owners[id] == 0
  @post (_owners[id] != 0) -> __return == true
 */
function _checkAndClear(address from, uint256 id) internal returns(bool) {
    uint256 owner = _owners[id];
    if (owner != 0) {
        require(address(owner) == from, "not owner");
        _owners[id] = 0;
        return true;
    }
    return false;
}

//@CTK FAIL NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK _regroup_require
  @tag assume_completion
  @post (x % size == 0) /\ (y % size == 0)
  @post (x <= GRID_SIZE - size) /\ (y <= GRID_SIZE - size)
  @post (size == 1 \/ size == 3 \/ size == 6 \/ size == 12 \/ size == 24)
 */
function _regroup(address from, address to, uint256 size, uint256 x, uint256 y) internal
     {
    require(x % size == 0 && y % size == 0, "Invalid coordinates");
    require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");

    if (size == 3) {
        _regroup3x3(from, to, x, y, true);
    } else if (size == 6) {
        _regroup6x6(from, to, x, y, true);
    } else if (size == 12) {
        _regroup12x12(from, to, x, y, true);
    } else if (size == 24) {
        _regroup24x24(from, to, x, y, true);
    } else {
        require(false, "Invalid size");
    }
}

//@CTK FAIL NO_OVERFLOW
```

```
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@*CTK _regroup3x3_require
      @tag assume_completion
      @pre GRID_SIZE == 408
      @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))
     */
    function _regroup3x3(address from, address to, uint256 x, uint256 y, bool set) internal
        returns (bool) {
        uint256 id = x + y * GRID_SIZE;
        uint256 quadId = LAYER_3x3 + id;
        bool ownerOfAll = true;
        for (uint256 xi = x; xi < x+3; xi++) {
            for (uint256 yi = y; yi < y+3; yi++) {
                ownerOfAll = _checkAndClear(from, xi + yi * GRID_SIZE) && ownerOfAll;
            }
        }
        if(set) {
            if(!ownerOfAll) {
                require(
                    _owners[quadId] == 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),
                    "not owner of all sub quads nor parent quads"
                );
            }
            _owners[quadId] = uint256(to);
            return true;
        }
        return ownerOfAll;
    }

    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    function _regroup6x6(address from, address to, uint256 x, uint256 y, bool set) internal
        returns (bool) {
        uint256 id = x + y * GRID_SIZE;
        uint256 quadId = LAYER_6x6 + id;
        bool ownerOfAll = true;
        for (uint256 xi = x; xi < x+6; xi += 3) {
            for (uint256 yi = y; yi < y+6; yi += 3) {
                bool ownAllIndividual = _regroup3x3(from, to, xi, yi, false);
                uint256 id3x3 = LAYER_3x3 + xi + yi * GRID_SIZE;
                uint256 owner3x3 = _owners[id3x3];
                if (owner3x3 != 0) {
                    if(!ownAllIndividual) {
                        require(owner3x3 == uint256(from), "not owner of 3x3 quad");
                    }
                    _owners[id3x3] = 0;
```

```
            }
            ownerOfAll = (ownAllIndividual || owner3x3 != 0) && ownerOfAll;
        }
    }
    if(set) {
        if(!ownerOfAll) {
            require(
                _owners[quadId] == 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),
                "not owner of all sub quads nor parent quads"
            );
        }
        _owners[quadId] = uint256(to);
        return true;
    }
    return ownerOfAll;
}

//@CTK FAIL NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
function _regroup12x12(address from, address to, uint256 x, uint256 y, bool set) internal returns (bool) {
    uint256 id = x + y * GRID_SIZE;
    uint256 quadId = LAYER_12x12 + id;
    bool ownerOfAll = true;
    for (uint256 xi = x; xi < x+12; xi += 6) {
        for (uint256 yi = y; yi < y+12; yi += 6) {
            bool ownAllIndividual = _regroup6x6(from, to, xi, yi, false);
            uint256 id6x6 = LAYER_6x6 + xi + yi * GRID_SIZE;
            uint256 owner6x6 = _owners[id6x6];
            if (owner6x6 != 0) {
                if(!ownAllIndividual) {
                    require(owner6x6 == uint256(from), "not owner of 6x6 quad");
                }
                _owners[id6x6] = 0;
            }
            ownerOfAll = (ownAllIndividual || owner6x6 != 0) && ownerOfAll;
        }
    }
    if(set) {
        if(!ownerOfAll) {
            require(
                _owners[quadId] == uint256(from) ||
                _owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE] == uint256(from),
                "not owner of all sub quads nor parent quads"
            );
        }
        _owners[quadId] = uint256(to);
        return true;
    }
    return ownerOfAll;
}

```

```
    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    function _regroup24x24(address from, address to, uint256 x, uint256 y, bool set)
        internal returns (bool) {
        uint256 id = x + y * GRID_SIZE;
        uint256 quadId = LAYER_24x24 + id;
        bool ownerOfAll = true;
        for (uint256 xi = x; xi < x+24; xi += 12) {
            for (uint256 yi = y; yi < y+24; yi += 12) {
                bool ownAllIndividual = _regroup12x12(from, to, xi, yi, false);
                uint256 id12x12 = LAYER_12x12 + xi + yi * GRID_SIZE;
                uint256 owner12x12 = _owners[id12x12];
                if (owner12x12 != 0) {
                    if(!ownAllIndividual) {
                        require(owner12x12 == uint256(from), "not owner of 12x12 quad");
                    }
                    _owners[id12x12] = 0;
                }
                ownerOfAll = (ownAllIndividual || owner12x12 != 0) && ownerOfAll;
            }
        }
        if(set) {
            if(!ownerOfAll) {
                require(
                    _owners[quadId] == uint256(from),
                    "not owner of all sub quads not parent quad"
                );
            }
            _owners[quadId] = uint256(to);
            return true;
        }
        return ownerOfAll || _owners[quadId] == uint256(from);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK FAIL "_ownerOf"
      @pre GRID_SIZE == 408
      @pre (id & LAYER) == 0
      @post (_owners[id] != 0) -> (__return == address(_owners[id]))
     */
    function _ownerOf(uint256 id) internal view returns (address) {
        require(id & LAYER == 0, "Invalid token id");
        uint256 x = id % GRID_SIZE;
        uint256 y = id / GRID_SIZE;
        uint256 owner1x1 = _owners[id];

        if (owner1x1 != 0) {
            return address(owner1x1); // cast to zero
        } else {
            address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) *
                GRID_SIZE]);
            if (owner3x3 != address(0)) {
                return owner3x3;
            } else {
                address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) *
```

```
                GRID_SIZE]);
            if (owner6x6 != address(0)) {
                return owner6x6;
            } else {
                address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE]);
                if (owner12x12 != address(0)) {
                    return owner12x12;
                } else {
                    return address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE]);
                }
            }
        }
    }
}

//@*CTK FAIL NO_OVERFLOW
//@*CTK NO_BUF_OVERFLOW
//@*CTK FAIL NO_ASF
/*@*CTK _ownerAndOperatorEnabledOf
  @pre GRID_SIZE == 408
  @pre (id & LAYER) == 0
  @post owner == address(_owners[id])
  @post operatorEnabled == ((_owners[id] / 2**255) == 1)
 */
function _ownerAndOperatorEnabledOf(uint256 id) internal view returns (address owner, bool operatorEnabled) {
    require(id & LAYER == 0, "Invalid token id");
    uint256 x = id % GRID_SIZE;
    uint256 y = id / GRID_SIZE;
    uint256 owner1x1 = _owners[id];

    if (owner1x1 != 0) {
        owner = address(owner1x1);
        operatorEnabled = (owner1x1 / 2**255) == 1;
    } else {
        address owner3x3 = address(_owners[LAYER_3x3 + (x/3) * 3 + ((y/3) * 3) * GRID_SIZE]);
        if (owner3x3 != address(0)) {
            owner = owner3x3;
            operatorEnabled = false;
        } else {
            address owner6x6 = address(_owners[LAYER_6x6 + (x/6) * 6 + ((y/6) * 6) * GRID_SIZE]);
            if (owner6x6 != address(0)) {
                owner = owner6x6;
                operatorEnabled = false;
            } else {
                address owner12x12 = address(_owners[LAYER_12x12 + (x/12) * 12 + ((y/12) * 12) * GRID_SIZE]);
                if (owner12x12 != address(0)) {
                    owner = owner12x12;
                    operatorEnabled = false;
                } else {
                    owner = address(_owners[LAYER_24x24 + (x/24) * 24 + ((y/24) * 24) * GRID_SIZE]);
                    operatorEnabled = false;
```

```
                    }
                }
            }
        }
    }

}
```