CertiK Audit Report For Paxos

Request Date: 2019-05-29
Revision Date: 2019-06-14
Platform Name: Ethereum

CERTIK

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

About CertiK

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

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

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 1.4B in assets.

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

Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by Paxos. This audit was conducted to discover issues and vulnerabilities in the source code of Paxos's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

For every issue found, CertiK categorizes them into 3 buckets based on its risk level:

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain conditions, or it could affect the security standard by lost of access control.

Low

The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

Testing Summary

Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

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

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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Scope of Work

Paxos invited CertiK to audit their soon to be released token based smart contracts with technical tools like formal verification to ensure the security and correctness of Blockchain Smart Contracts.

The goal of this audit is to review PAXG solidity implementation for its business model, detect potential security vulnerabilities, understand its general design and architecture, and uncover bugs that could compromise the software in the production environment.

In addition to manually checking for errors, CertiK employed our proprietary Formal Verification process, together with static analysis tools to mathematically ensure the entire code logic at-scale works as intended.

The audited source code **SHA-256 Checksum**:

- **Migrations.sol**
  `076805decaf87ca273550bfb4b7f7ec53a66266f5bfa74704771654e0404b2ba`
- **PAXGImplementation.sol**
  `c031eeeed47a61d8425589fdfe2060a9644d33b0165bc7fa8d516cbaef3933a4`

Methodology

CertiK audits projects using three different working layers to maximize security by layering perspectives and tools. These include:

1. CertiK *Formal Verification*
2. *Manual Reviews* by smart contract experts
3. *Static Analysis*

By using these three approaches in tandem, CertiK is able to observe code on both a holistic, all-encompassing approach as well as line-by-line specific analysis.

Documentation

CertiK used the following sources of truth about how PAXG should work:

1. Client's Project README Files.
2. Previous Audit-Reports.
3. Test Scenarios (PAXG provided high coverage unit tests to simulate possible scenarios).
4. Paxos Github Code Base.
5. Paxos Standard Token @ 0x8e870d67f660d95d5be530380d0ec0bd388289e1.

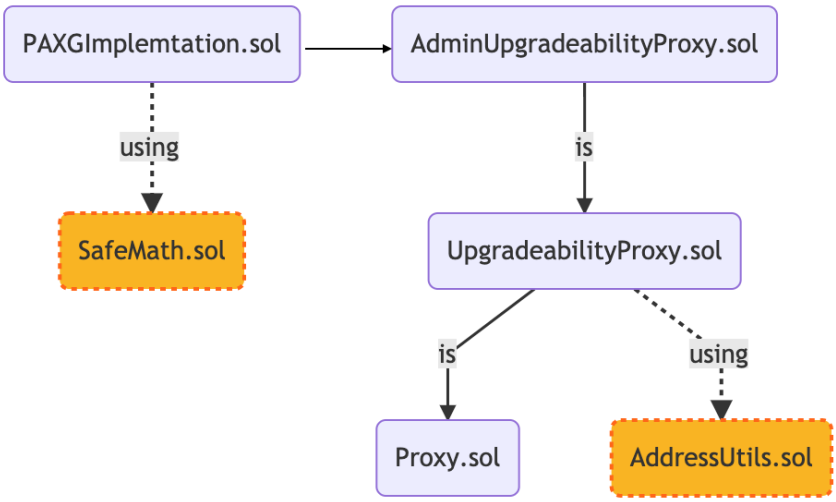
All listed sources act as a specification. If we discovered inconsistencies within the actual code behavior, we consulted with the Paxos team for further discussion and confirmation.

Summary

The Paxos Standard Token(PAX), is one of the few regulated stable-coins in the current cryptocurrency market. As of 04 June, 2019 PAX had a market capitalization of 169 million USD, and 24 Hour Volume of 177 million USD.

With such an impressive background, Paxos is highly regarded across the community, making it one of the premiere digital assets in the space. As Paxos continues to scale, it prepared a new ERC20 token called 'PAXG' to be implemented for upcoming business expansion, challenges, and opportunities.

As compared to the current PAX smart contract, PAXG contract introduces an additional fee logic, where the fee rate can be controlled by the feeController role. The PAXG contract also introduces support for delegate transfer, which enables transfer by transactor on behalf of the investor with consent from the investor (in the form of signature, where the signed message is associated with transaction details such as recipient, value, etc). Additionally, the transactor can optionally charge the investor a certain amount of service fee.



Such a project requires the utmost in security protocols. To that end, PAXG ensures the use of sequence numbers and deadlines to prevent the signature from being abused. The off-chain transaction signing and delegate related functions are currently available for whitelisted actors and in `beta` mode.

Recommendations

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

PAXGImplementation.sol

1. `betaDelegateTransfer()`: Recommend checking signature field $s \in (0, \text{secp256k1n}/2+1)$ and field $v \in 27, 28$.

- [**Paxos - updated**]: The signature check condition is updated as recommended.

2. In `_betaDelegatedTransfer()` the block.number is used as deadline. One caveat of such approach is that block.number can be manipulated by miner, thus we recommend using timestamp as deadline instead. According to the Ethereum Yellow Paper, the block.timestamp of blocks must be newer than its parent block, thus manipulation of block.timestamp is less likely to happen and makes the deadline in the signed message more reliable. Reference: The 15-Second Rule.

   - [**Paxos - confirmed**]: We chose to include a deadline in terms of a block number following vbuterin's proposal entitled "Layer 2 gas payment abstraction" on Eth Research. The suggested usage is for the delegate/service provider to provide a good default when they provide the gas price, such as 24 hours in blocks (roughly 6000 at 15s/block). This incentivizes the delegate to process the transaction since the opportunity expires and allows for a better default than infinite transaction livelihood.

3. Recommend extracting EIP-712 related logic in `_betaDelegatedTransfer` and `betaDelegatedTransfer` into separate internal functions or a separate library, keeping the responsibility of `_betaDelegatedTransfer` simple.

   - [**Paxos - confirmed**]: We believe it is easier to reason about `_betaDelegatedTransfer` is this logic is kept in line with the rest of the function, especially after refactoring out the transfer logic.

4. Unused public state variable feeDecimals could be removed if not for explanatory purpose.

   - [**Paxos - confirmed**]: The unused public state variable `feeDecimals` is only for the explanatory purpose of how many decimals the fees are calculated in. We have added additional comments in the contract to clarify this.

5. Recommend extracting fee reduction logic in `transfer()`, `transferFrom()`, `_betaDelegatedTransfer()` into an internal transfer function to avoid redundancy.

   - [**Paxos - updated**]: A new function `_transfer()` now handles this logic as suggested.

6. Inconsistent parameter style of public methods `betaDelegatedTransfer` and `betaDelegatedTransferBatch`. It would be better for the client to provide additional info on the design consideration.

   - [**Paxos - confirmed**]: Solidity 0.4 does have very good support for arrays of objects and has neither support for fixed-size arrays or nested arrays, so we were unable to match the parameter style. We did, however, want to keep things simple for the more common singular delegated transfer so it only requires the signature.

Best practice

Smart contract development requires a particular engineering mindset. A failure in the initial construction can be catastrophic, and changing the project after the fact can be exceedingly difficult.

To ensure success and to avoid the challenges above smart contracts should here to best practices at their conception. Below, we summarized a checklist of key points that help to indicate a high overall quality of the Paxos project.

Solidity Protocol

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

Privilege Control

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

Documentation

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

Testing

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

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

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

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File PAXGImplementation.sol

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

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

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 }
```

Post environment

```
56
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

initialize

14, Jun 2019
197.27 ms

Line 170-182 in File PAXGImplementation.sol

```
170   /*@CTK initialize
171     @tag assume_completion
172     @post !initialized
173     @post __post.owner == msg.sender
174     @post __post.proposedOwner == address(0)
175     @post __post.regulatoryComplianceRole == address(0)
176     @post __post.totalSupply_ == 0
177     @post __post.supplyController == msg.sender
178     @post __post.feeRate == 0
179     @post __post.feeController == msg.sender
180     @post __post.feeRecipient == msg.sender
181     @post __post.initialized
182    */
```

Line 183-195 in File PAXGImplementation.sol

```
183   function initialize() public {
184       require(!initialized, "already initialized");
185       owner = msg.sender;
186       proposedOwner = address(0);
187       regulatoryComplianceRole = address(0);
188       totalSupply_ = 0;
189       supplyController = msg.sender;
190       feeRate = 0;
191       feeController = msg.sender;
192       feeRecipient = msg.sender;
193       initializeDomainSeparator();
194       initialized = true;
195   }
```

The code meets the specification.

Formal Verification Request 2

totalSupply

14, Jun 2019
5.94 ms

Line 225-227 in File PAXGImplementation.sol

```
225   /*@CTK totalSupply
226     @post __return == totalSupply_
227    */
```

Line 228-230 in File PAXGImplementation.sol

```
228   function totalSupply() public view returns (uint256) {
229       return totalSupply_;
230   }
```

The code meets the specification.

Formal Verification Request 3

transfer

14, Jun 2019
700.9 ms

Line 239-246 in File PAXGImplementation.sol

```
/*@CTK transfer
  @tag assume_completion
  @pre _to != msg.sender && _to != feeRecipient && msg.sender != feeRecipient
  @post !paused
  @post _to != address(0)
  @post !frozen[_to] && !frozen[msg.sender]
  @post _value <= balances[msg.sender]
 */
```

Line 247-254 in File PAXGImplementation.sol

```
function transfer(address _to, uint256 _value) public whenNotPaused returns (bool)
    {
    require(_to != address(0), "cannot transfer to address zero");
    require(!frozen[_to] && !frozen[msg.sender], "address frozen");
    require(_value <= balances[msg.sender], "insufficient funds");

    _transfer(msg.sender, _to, _value);
    return true;
}
```

The code meets the specification.

Formal Verification Request 4

balanceOf

14, Jun 2019
5.56 ms

Line 261-263 in File PAXGImplementation.sol

```
/*@CTK balanceOf
  @post __return == balances[_addr]
 */
```

Line 264-266 in File PAXGImplementation.sol

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

The code meets the specification.

Formal Verification Request 5

transferFrom

14, Jun 2019
604.34 ms

Line 276-284 in File PAXGImplementation.sol

```
276    /*@CTK transferFrom
277      @tag assume_completion
278      @pre _to != _from && _to != feeRecipient && _from != feeRecipient
279      @post !paused
280      @post _to != address(0)
281      @post !frozen[_to] && !frozen[_from] && !frozen[msg.sender]
282      @post _value <= balances[_from]
283      @post _value <= allowed[_from][msg.sender]
284     */
```

Line 285-303 in File PAXGImplementation.sol

```
285    function transferFrom(
286        address _from,
287        address _to,
288        uint256 _value
289    )
290    public
291    whenNotPaused
292    returns (bool)
293    {
294        require(_to != address(0), "cannot transfer to address zero");
295        require(!frozen[_to] && !frozen[_from] && !frozen[msg.sender], "address frozen"
           );
296        require(_value <= balances[_from], "insufficient funds");
297        require(_value <= allowed[_from][msg.sender], "insufficient allowance");
298
299        allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
300        _transfer(_from, _to, _value);
301
302        return true;
303    }
```

The code meets the specification.

Formal Verification Request 6

approve

14, Jun 2019
64.58 ms

Line 314-319 in File PAXGImplementation.sol

```
314    /*@CTK approve
315      @tag assume_completion
316      @post !paused
317      @post !frozen[_spender] && !frozen[msg.sender]
318      @post __post.allowed[msg.sender][_spender] == _value
```

```
319     */
```

Line 320-325 in File PAXGImplementation.sol

```
320     function approve(address _spender, uint256 _value) public whenNotPaused returns (
            bool) {
321         require(!frozen[_spender] && !frozen[msg.sender], "address frozen");
322         allowed[msg.sender][_spender] = _value;
323         emit Approval(msg.sender, _spender, _value);
324         return true;
325     }
```

The code meets the specification.

Formal Verification Request 7

allowance

14, Jun 2019
6.81 ms

Line 333-335 in File PAXGImplementation.sol

```
333     /*@CTK allowance
334       @post __return == allowed[_owner][_spender]
335      */
```

Line 336-345 in File PAXGImplementation.sol

```
336     function allowance(
337         address _owner,
338         address _spender
339     )
340     public
341     view
342     returns (uint256)
343     {
344         return allowed[_owner][_spender];
345     }
```

The code meets the specification.

Formal Verification Request 8

_transfer

14, Jun 2019
69077.1 ms

Line 347-356 in File PAXGImplementation.sol

```
347     /*@CTK "_transfer"
348       @tag assume_completion
349       @pre feeParts > 0
350       @post _value <= balances[_from]
351       @post (_from == _to && feeRecipient == _from) -> (__post.balances[_to] ==
              balances[_to])
```

```
352     @post (_from == _to && feeRecipient != _from) -> (__post.balances[_to] ==
            balances[_to] - _value * feeRate / feeParts) && (__post.balances[feeRecipient
            ] == balances[feeRecipient] + _value * feeRate / feeParts)
353     @post (_from != _to && feeRecipient == _from) -> (__post.balances[_to] ==
            balances[_to] + _value - _value * feeRate / feeParts) && (__post.balances[
            _from] == balances[_from] - _value + _value * feeRate / feeParts)
354     @post (_from != _to && feeRecipient == _to) -> (__post.balances[_to] == balances
            [_to] + _value) && (__post.balances[_from] == balances[_from] - _value)
355     @post (_from != _to && feeRecipient != _from && feeRecipient != _to) -> (__post.
            balances[_to] == balances[_to] + _value - _value * feeRate / feeParts) && (
            __post.balances[_from] == balances[_from] - _value) && (__post.balances[
            feeRecipient] == balances[feeRecipient] + _value * feeRate / feeParts)
356    */
```

Line 357-370 in File PAXGImplementation.sol

```
357    function _transfer(address _from, address _to, uint256 _value) internal returns (
           uint256) {
358        uint256 _fee = getFeeFor(_value);
359        uint256 _principle = _value.sub(_fee);
360        balances[_from] = balances[_from].sub(_value);
361        balances[_to] = balances[_to].add(_principle);
362        emit Transfer(_from, _to, _principle);
363        emit Transfer(_from, feeRecipient, _fee);
364        if (_fee > 0) {
365            balances[feeRecipient] = balances[feeRecipient].add(_fee);
366            emit FeeCollected(_from, feeRecipient, _fee);
367        }
368
369        return _principle;
370    }
```

✅ The code meets the specification.

Formal Verification Request 9

proposeOwner

📅 14, Jun 2019
⏱ 50.36 ms

Line 386-391 in File PAXGImplementation.sol

```
386    /*@CTK proposeOwner
387      @tag assume_completion
388      @post _proposedOwner != address(0)
389      @post _proposedOwner != msg.sender
390      @post __post.proposedOwner == _proposedOwner
391     */
```

Line 392-397 in File PAXGImplementation.sol

```
392    function proposeOwner(address _proposedOwner) public onlyOwner {
393        require(_proposedOwner != address(0), "cannot transfer ownership to address
               zero");
394        require(msg.sender != _proposedOwner, "caller already is owner");
395        proposedOwner = _proposedOwner;
396        emit OwnershipTransferProposed(owner, proposedOwner);
397    }
```

The code meets the specification.

Formal Verification Request 10

proposeOwner

14, Jun 2019
48.09 ms

Line 402-407 in File PAXGImplementation.sol

```
402    /*@CTK proposeOwner
403      @tag assume_completion
404      @post msg.sender == proposedOwner || msg.sender == owner
405      @post proposedOwner != address(0)
406      @post __post.proposedOwner == address(0)
407     */
```

Line 408-414 in File PAXGImplementation.sol

```
408    function disregardProposeOwner() public {
409        require(msg.sender == proposedOwner || msg.sender == owner, "only proposedOwner
               or owner");
410        require(proposedOwner != address(0), "can only disregard a proposed owner that
               was previously set");
411        address _oldProposedOwner = proposedOwner;
412        proposedOwner = address(0);
413        emit OwnershipTransferDisregarded(_oldProposedOwner);
414    }
```

The code meets the specification.

Formal Verification Request 11

claimOwnership

14, Jun 2019
36.42 ms

Line 419-424 in File PAXGImplementation.sol

```
419    /*@CTK claimOwnership
420      @tag assume_completion
421      @post msg.sender == proposedOwner
422      @post __post.owner == proposedOwner
423      @post __post.proposedOwner == address(0)
424     */
```

Line 425-431 in File PAXGImplementation.sol

```
425    function claimOwnership() public {
426        require(msg.sender == proposedOwner, "onlyProposedOwner");
427        address _oldOwner = owner;
428        owner = proposedOwner;
429        proposedOwner = address(0);
430        emit OwnershipTransferred(_oldOwner, owner);
431    }
```

The code meets the specification.

Formal Verification Request 12

reclaimPAXG

14, Jun 2019
139.11 ms

Line 438-444 in File PAXGImplementation.sol

```
/*@CTK reclaimPAXG
  @tag assume_completion
  @pre owner != this
  @post owner == msg.sender
  @post __post.balances[this] == 0
  @post __post.balances[owner] == balances[owner] + balances[this]
 */
```

Line 445-450 in File PAXGImplementation.sol

```
function reclaimPAXG() external onlyOwner {
    uint256 _balance = balances[this];
    balances[this] = 0;
    balances[owner] = balances[owner].add(_balance);
    emit Transfer(this, owner, _balance);
}
```

The code meets the specification.

Formal Verification Request 13

pause

14, Jun 2019
35.39 ms

Line 465-469 in File PAXGImplementation.sol

```
/*@CTK pause
  @tag assume_completion
  @post owner == msg.sender
  @post __post.paused
 */
```

Line 470-474 in File PAXGImplementation.sol

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

The code meets the specification.

Formal Verification Request 14

unpause

14, Jun 2019
33.62 ms

Line 479-483 in File PAXGImplementation.sol

```
/*@CTK unpause
  @tag assume_completion
  @post owner == msg.sender
  @post !__post.paused
 */
```

Line 484-488 in File PAXGImplementation.sol

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

The code meets the specification.

Formal Verification Request 15

setRegulatoryComplianceRole

14, Jun 2019
27.88 ms

Line 496-500 in File PAXGImplementation.sol

```
/*@CTK setRegulatoryComplianceRole
  @tag assume_completion
  @post msg.sender == regulatoryComplianceRole || msg.sender == owner
  @post __post.regulatoryComplianceRole == _newRegulatoryComplianceRole
 */
```

Line 501-505 in File PAXGImplementation.sol

```
function setRegulatoryComplianceRole(address _newRegulatoryComplianceRole) public
    {
    require(msg.sender == regulatoryComplianceRole || msg.sender == owner, "only
        regulatoryComplianceRole or Owner");
    emit RegulatoryComplianceRoleSet(regulatoryComplianceRole,
        _newRegulatoryComplianceRole);
    regulatoryComplianceRole = _newRegulatoryComplianceRole;
}
```

The code meets the specification.

Formal Verification Request 16

freeze

14, Jun 2019
36.43 ms

Line 516-521 in File PAXGImplementation.sol

```
516   /*@CTK freeze
517     @tag assume_completion
518     @post msg.sender == regulatoryComplianceRole
519     @post !frozen[_addr]
520     @post __post.frozen[_addr]
521    */
```

Line 522-526 in File PAXGImplementation.sol

```
522   function freeze(address _addr) public onlyRegulatoryComplianceRole {
523       require(!frozen[_addr], "address already frozen");
524       frozen[_addr] = true;
525       emit AddressFrozen(_addr);
526   }
```

The code meets the specification.

Formal Verification Request 17

unfreeze

14, Jun 2019
36.28 ms

Line 532-537 in File PAXGImplementation.sol

```
532   /*@CTK unfreeze
533     @tag assume_completion
534     @post msg.sender == regulatoryComplianceRole
535     @post frozen[_addr]
536     @post !__post.frozen[_addr]
537    */
```

Line 538-542 in File PAXGImplementation.sol

```
538   function unfreeze(address _addr) public onlyRegulatoryComplianceRole {
539       require(frozen[_addr], "address already unfrozen");
540       frozen[_addr] = false;
541       emit AddressUnfrozen(_addr);
542   }
```

The code meets the specification.

Formal Verification Request 18

isFrozen

14, Jun 2019
5.79 ms

Line 571-573 in File PAXGImplementation.sol

```
571   /*@CTK isFrozen
572     @post __return == frozen[_addr]
573    */
```

Line 574-576 in File PAXGImplementation.sol

```
574 function isFrozen(address _addr) public view returns (bool) {
575     return frozen[_addr];
576 }
```

The code meets the specification.

Formal Verification Request 19

setSupplyController

14, Jun 2019
40.81 ms

Line 584-589 in File PAXGImplementation.sol

```
584 /*@CTK setSupplyController
585   @tag assume_completion
586   @post msg.sender == supplyController || msg.sender == owner
587   @post _newSupplyController != address(0)
588   @post __post.supplyController == _newSupplyController
589  */
```

Line 590-595 in File PAXGImplementation.sol

```
590 function setSupplyController(address _newSupplyController) public {
591     require(msg.sender == supplyController || msg.sender == owner, "only
            SupplyController or Owner");
592     require(_newSupplyController != address(0), "cannot set supply controller to
            address zero");
593     emit SupplyControllerSet(supplyController, _newSupplyController);
594     supplyController = _newSupplyController;
595 }
```

The code meets the specification.

Formal Verification Request 20

nextSeqOf

14, Jun 2019
5.79 ms

Line 652-654 in File PAXGImplementation.sol

```
652 /*@CTK nextSeqOf
653   @post __return == nextSeqs[target]
654  */
```

Line 655-657 in File PAXGImplementation.sol

```
655 function nextSeqOf(address target) public view returns (uint256) {
656     return nextSeqs[target];
657 }
```

The code meets the specification.

Formal Verification Request 21

isWhitelistedBetaDelegate

14, Jun 2019
5.83 ms

Line 772-774 in File PAXGImplementation.sol

```
/*@CTK isWhitelistedBetaDelegate
  @post __return == betaDelegateWhitelist[_addr]
 */
```

Line 775-777 in File PAXGImplementation.sol

```
function isWhitelistedBetaDelegate(address _addr) public view returns (bool) {
    return betaDelegateWhitelist[_addr];
}
```

The code meets the specification.

Formal Verification Request 22

setBetaDelegateWhitelister

14, Jun 2019
26.9 ms

Line 783-787 in File PAXGImplementation.sol

```
/*@CTK setBetaDelegateWhitelister
  @tag assume_completion
  @post msg.sender == betaDelegateWhitelister || msg.sender == owner
  @post __post.betaDelegateWhitelister == _newWhitelister
 */
```

Line 788-792 in File PAXGImplementation.sol

```
function setBetaDelegateWhitelister(address _newWhitelister) public {
    require(msg.sender == betaDelegateWhitelister || msg.sender == owner, "only
        Whitelister or Owner");
    betaDelegateWhitelister = _newWhitelister;
    emit BetaDelegateWhitelisterSet(betaDelegateWhitelister, _newWhitelister);
}
```

The code meets the specification.

Formal Verification Request 23

whitelistBetaDelegate

14, Jun 2019
39.67 ms

Line 803-808 in File PAXGImplementation.sol

```
/*@CTK whitelistBetaDelegate
  @tag assume_completion
  @post msg.sender == betaDelegateWhitelister
  @post !betaDelegateWhitelist[_addr]
  @post __post.betaDelegateWhitelist[_addr]
 */
```

Line 809-813 in File PAXGImplementation.sol

```
function whitelistBetaDelegate(address _addr) public onlyBetaDelegateWhitelister {
    require(!betaDelegateWhitelist[_addr], "delegate already whitelisted");
    betaDelegateWhitelist[_addr] = true;
    emit BetaDelegateWhitelisted(_addr);
}
```

The code meets the specification.

Formal Verification Request 24

unwhitelistBetaDelegate

14, Jun 2019
38.48 ms

Line 819-824 in File PAXGImplementation.sol

```
/*@CTK unwhitelistBetaDelegate
  @tag assume_completion
  @post msg.sender == betaDelegateWhitelister
  @post betaDelegateWhitelist[_addr]
  @post !__post.betaDelegateWhitelist[_addr]
 */
```

Line 825-829 in File PAXGImplementation.sol

```
function unwhitelistBetaDelegate(address _addr) public onlyBetaDelegateWhitelister
    {
    require(betaDelegateWhitelist[_addr], "delegate not whitelisted");
    betaDelegateWhitelist[_addr] = false;
    emit BetaDelegateUnwhitelisted(_addr);
}
```

The code meets the specification.

Formal Verification Request 25

setFeeController

14, Jun 2019
46.35 ms

Line 837-842 in File PAXGImplementation.sol

```
/*@CTK setFeeController
  @tag assume_completion
  @post msg.sender == feeController || msg.sender == owner
  @post _newFeeController != address(0)
```

```
841     @post __post.feeController == _newFeeController
842    */
```

Line 843-849 in File PAXGImplementation.sol

```
843    function setFeeController(address _newFeeController) public {
844        require(msg.sender == feeController || msg.sender == owner, "only FeeController
               or Owner");
845        require(_newFeeController != address(0), "cannot set fee controller to address
               zero");
846        address _oldFeeController = feeController;
847        feeController = _newFeeController;
848        emit FeeControllerSet(_oldFeeController, feeController);
849    }
```

The code meets the specification.

Formal Verification Request 26

setFeeRecipient

14, Jun 2019
41.24 ms

Line 860-865 in File PAXGImplementation.sol

```
860    /*@CTK setFeeRecipient
861      @tag assume_completion
862      @post msg.sender == feeController
863      @post _newFeeRecipient != address(0)
864      @post __post.feeRecipient == _newFeeRecipient
865     */
```

Line 866-871 in File PAXGImplementation.sol

```
866    function setFeeRecipient(address _newFeeRecipient) public onlyFeeController {
867        require(_newFeeRecipient != address(0), "cannot set fee recipient to address
               zero");
868        address _oldFeeRecipient = feeRecipient;
869        feeRecipient = _newFeeRecipient;
870        emit FeeRecipientSet(_oldFeeRecipient, feeRecipient);
871    }
```

The code meets the specification.

Formal Verification Request 27

setFeeRate

14, Jun 2019
41.79 ms

Line 877-882 in File PAXGImplementation.sol

```
877    /*@CTK setFeeRate
878      @tag assume_completion
879      @post msg.sender == feeController
```

```
     @post _newFeeRate <= feeParts
     @post __post.feeRate == _newFeeRate
    */
```

Line 883-888 in File PAXGImplementation.sol

```
   function setFeeRate(uint256 _newFeeRate) public onlyFeeController {
       require(_newFeeRate <= feeParts, "cannot set fee rate above 100%");
       uint256 _oldFeeRate = feeRate;
       feeRate = _newFeeRate;
       emit FeeRateSet(_oldFeeRate, feeRate);
   }
```

The code meets the specification.

Formal Verification Request 28

getFeeFor

14, Jun 2019
480.08 ms

Line 895-899 in File PAXGImplementation.sol

```
   /*@CTK getFeeFor
     @tag assume_completion
     @post (feeRate == 0) -> (__return == 0)
     @post (feeRate != 0) -> (__return == _value * feeRate / feeParts)
    */
```

Line 900-906 in File PAXGImplementation.sol

```
   function getFeeFor(uint256 _value) public view returns (uint256) {
       if (feeRate == 0) {
           return 0;
       }

       return _value.mul(feeRate).div(feeParts);
   }
```

The code meets the specification.

Source Code with CertiK Labels

File PAXGImplementation.sol

```
pragma solidity ^0.4.24;


import "./zeppelin/SafeMath.sol";


/**
 * @title PAXGImplementation
 * @dev this contract is a Pausable ERC20 token with Burn and Mint
 * controlled by a central SupplyController. By implementing PaxosImplementation
 * this contract also includes external methods for setting
 * a new implementation contract for the Proxy.
 * NOTE: The storage defined here will actually be held in the Proxy
 * contract and all calls to this contract should be made through
 * the proxy, including admin actions done as owner or supplyController.
 * Any call to transfer against this contract should fail
 * with insufficient funds since no tokens will be issued there.
 */
contract PAXGImplementation {

    /**
     * MATH
     */

    using SafeMath for uint256;

    /**
     * DATA
     */

    // INITIALIZATION DATA
    bool private initialized = false;

    // ERC20 BASIC DATA
    mapping(address => uint256) internal balances;
    uint256 internal totalSupply_;
    string public constant name = "Paxos Gold"; // solium-disable-line
    string public constant symbol = "PAXG"; // solium-disable-line uppercase
    uint8 public constant decimals = 18; // solium-disable-line uppercase

    // ERC20 DATA
    mapping(address => mapping(address => uint256)) internal allowed;

    // OWNER DATA
    address public owner;
    address public proposedOwner;

    // PAUSABILITY DATA
    bool public paused = false;

    // REGULATORY COMPLIANCE DATA
    address public regulatoryComplianceRole;
    mapping(address => bool) internal frozen;

```

```
    // SUPPLY CONTROL DATA
    address public supplyController;

    // DELEGATED TRANSFER DATA
    address public betaDelegateWhitelister;
    mapping(address => bool) internal betaDelegateWhitelist;
    mapping(address => uint256) internal nextSeqs;
    // EIP191 header for EIP712 prefix
    string constant internal EIP191_HEADER = "\x19\x01";
    // Hash of the EIP712 Domain Separator Schema
    bytes32 constant internal EIP712_DOMAIN_SEPARATOR_SCHEMA_HASH = keccak256(
        "EIP712Domain(string name,address verifyingContract)"
    );
    bytes32 constant internal EIP712_DELEGATED_TRANSFER_SCHEMA_HASH = keccak256(
        "BetaDelegatedTransfer(address to,uint256 value,uint256 serviceFee,uint256 seq,
            uint256 deadline)"
    );
    // Hash of the EIP712 Domain Separator data
    // solhint-disable-next-line var-name-mixedcase
    bytes32 public EIP712_DOMAIN_HASH;

    // FEE CONTROLLER DATA
    // feeRate is measured in 100th of a basis point (parts per 1,000,000)
    // ex: 200 fee parts = 0.02%
    uint8 public constant feeDecimals = 6;
    uint256 public constant feeParts = 1000000;
    uint256 public feeRate;
    address public feeController;
    address public feeRecipient;

    /**
     * EVENTS
     */

    // ERC20 BASIC EVENTS
    event Transfer(address indexed from, address indexed to, uint256 value);

    // ERC20 EVENTS
    event Approval(
        address indexed owner,
        address indexed spender,
        uint256 value
    );

    // OWNABLE EVENTS
    event OwnershipTransferProposed(
        address indexed currentOwner,
        address indexed proposedOwner
    );
    event OwnershipTransferDisregarded(
        address indexed oldProposedOwner
    );
    event OwnershipTransferred(
        address indexed oldOwner,
        address indexed newOwner
    );

    // PAUSABLE EVENTS
```

```
    event Pause();
    event Unpause();

    // REGULATORY COMPLIANCE EVENTS
    event AddressFrozen(address indexed addr);
    event AddressUnfrozen(address indexed addr);
    event FrozenAddressWiped(address indexed addr);
    event RegulatoryComplianceRoleSet (
        address indexed oldRegulatoryComplianceRole,
        address indexed newRegulatoryComplianceRole
    );

    // SUPPLY CONTROL EVENTS
    event SupplyIncreased(address indexed to, uint256 value);
    event SupplyDecreased(address indexed from, uint256 value);
    event SupplyControllerSet(
        address indexed oldSupplyController,
        address indexed newSupplyController
    );

    // DELEGATED TRANSFER EVENTS
    event BetaDelegatedTransfer(
        address indexed from, address indexed to, uint256 value, uint256 seq, uint256
            serviceFee
    );
    event BetaDelegateWhitelisterSet(
        address indexed oldWhitelister,
        address indexed newWhitelister
    );
    event BetaDelegateWhitelisted(address indexed newDelegate);
    event BetaDelegateUnwhitelisted(address indexed oldDelegate);

    // FEE CONTROLLER EVENTS
    event FeeCollected(address indexed from, address indexed to, uint256 value);
    event FeeRateSet(
        uint256 indexed oldFeeRate,
        uint256 indexed newFeeRate
    );
    event FeeControllerSet(
        address indexed oldFeeController,
        address indexed newFeeController
    );
    event FeeRecipientSet(
        address indexed oldFeeRecipient,
        address indexed newFeeRecipient
    );

    /**
     * FUNCTIONALITY
     */

    // INITIALIZATION FUNCTIONALITY

    /**
     * @dev sets 0 initial tokens, the owner, the supplyController,
     * the fee controller and fee recipient.
     * this serves as the constructor for the proxy but compiles to the
     * memory model of the Implementation contract.
```

```
     */
    /*@CTK initialize
      @tag assume_completion
      @post !initialized
      @post __post.owner == msg.sender
      @post __post.proposedOwner == address(0)
      @post __post.regulatoryComplianceRole == address(0)
      @post __post.totalSupply_ == 0
      @post __post.supplyController == msg.sender
      @post __post.feeRate == 0
      @post __post.feeController == msg.sender
      @post __post.feeRecipient == msg.sender
      @post __post.initialized
     */
    function initialize() public {
        require(!initialized, "already initialized");
        owner = msg.sender;
        proposedOwner = address(0);
        regulatoryComplianceRole = address(0);
        totalSupply_ = 0;
        supplyController = msg.sender;
        feeRate = 0;
        feeController = msg.sender;
        feeRecipient = msg.sender;
        initializeDomainSeparator();
        initialized = true;
    }

    /**
     * The constructor is used here to ensure that the implementation
     * contract is initialized. An uncontrolled implementation
     * contract might lead to misleading state
     * for users who accidentally interact with it.
     */
    constructor() public {
        initialize();
        pause();
    }

    /**
     * @dev To be called when upgrading the contract using upgradeAndCall to add
         delegated transfers
     */
    function initializeDomainSeparator() public {
        // hash the name context with the contract address
        EIP712_DOMAIN_HASH = keccak256(abi.encodePacked(// solium-disable-line
                EIP712_DOMAIN_SEPARATOR_SCHEMA_HASH,
                keccak256(bytes(name)),
                bytes32(address(this))
            ));
    }

    // ERC20 BASIC FUNCTIONALITY

    /**
    * @dev Total number of tokens in existence
    */
    /*@CTK totalSupply
```

```
    @post __return == totalSupply_
   */
  function totalSupply() public view returns (uint256) {
      return totalSupply_;
  }

  /**
  * @dev Transfer token to a specified address from msg.sender
  * Transfer additionally sends the fee to the fee controller
  * Note: the use of Safemath ensures that _value is nonnegative.
  * @param _to The address to transfer to.
  * @param _value The amount to be transferred.
  */
  /*@CTK transfer
    @tag assume_completion
    @pre _to != msg.sender && _to != feeRecipient && msg.sender != feeRecipient
    @post !paused
    @post _to != address(0)
    @post !frozen[_to] && !frozen[msg.sender]
    @post _value <= balances[msg.sender]
   */
  function transfer(address _to, uint256 _value) public whenNotPaused returns (bool)
      {
      require(_to != address(0), "cannot transfer to address zero");
      require(!frozen[_to] && !frozen[msg.sender], "address frozen");
      require(_value <= balances[msg.sender], "insufficient funds");

      _transfer(msg.sender, _to, _value);
      return true;
  }

  /**
  * @dev Gets the balance of the specified address.
  * @param _addr The address to query the the balance of.
  * @return An uint256 representing the amount owned by the passed address.
  */
  /*@CTK balanceOf
    @post __return == balances[_addr]
   */
  function balanceOf(address _addr) public view returns (uint256) {
      return balances[_addr];
  }

  // ERC20 FUNCTIONALITY

  /**
   * @dev Transfer tokens from one address to another
   * @param _from address The address which you want to send tokens from
   * @param _to address The address which you want to transfer to
   * @param _value uint256 the amount of tokens to be transferred
   */
  /*@CTK transferFrom
    @tag assume_completion
    @pre _to != _from && _to != feeRecipient && _from != feeRecipient
    @post !paused
    @post _to != address(0)
    @post !frozen[_to] && !frozen[_from] && !frozen[msg.sender]
    @post _value <= balances[_from]
```

```
     @post _value <= allowed[_from][msg.sender]
    */
    function transferFrom(
        address _from,
        address _to,
        uint256 _value
    )
    public
    whenNotPaused
    returns (bool)
    {
        require(_to != address(0), "cannot transfer to address zero");
        require(!frozen[_to] && !frozen[_from] && !frozen[msg.sender], "address frozen"
            );
        require(_value <= balances[_from], "insufficient funds");
        require(_value <= allowed[_from][msg.sender], "insufficient allowance");

        allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
        _transfer(_from, _to, _value);

        return true;
    }

    /**
     * @dev Approve the passed address to spend the specified amount of tokens on
         behalf of msg.sender.
     * Beware that changing an allowance with this method brings the risk that someone
          may use both the old
     * and the new allowance by unfortunate transaction ordering. One possible
         solution to mitigate this
     * race condition is to first reduce the spender's allowance to 0 and set the
         desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     * @param _spender The address which will spend the funds.
     * @param _value The amount of tokens to be spent.
     */
    /*@CTK approve
      @tag assume_completion
      @post !paused
      @post !frozen[_spender] && !frozen[msg.sender]
      @post __post.allowed[msg.sender][_spender] == _value
     */
    function approve(address _spender, uint256 _value) public whenNotPaused returns (
        bool) {
        require(!frozen[_spender] && !frozen[msg.sender], "address frozen");
        allowed[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;
    }

    /**
     * @dev Function to check the amount of tokens that an owner allowed to a spender.
     * @param _owner address The address which owns the funds.
     * @param _spender address The address which will spend the funds.
     * @return A uint256 specifying the amount of tokens still available for the
         spender.
     */
    /*@CTK allowance
```

```
     @post __return == allowed[_owner][_spender]
    */
    function allowance(
        address _owner,
        address _spender
    )
    public
    view
    returns (uint256)
    {
        return allowed[_owner][_spender];
    }

    /*@CTK "_transfer"
      @tag assume_completion
      @pre feeParts > 0
      @post _value <= balances[_from]
      @post (_from == _to && feeRecipient == _from) -> (__post.balances[_to] ==
          balances[_to])
      @post (_from == _to && feeRecipient != _from) -> (__post.balances[_to] ==
          balances[_to] - _value * feeRate / feeParts) && (__post.balances[feeRecipient
          ] == balances[feeRecipient] + _value * feeRate / feeParts)
      @post (_from != _to && feeRecipient == _from) -> (__post.balances[_to] ==
          balances[_to] + _value - _value * feeRate / feeParts) && (__post.balances[
          _from] == balances[_from] - _value + _value * feeRate / feeParts)
      @post (_from != _to && feeRecipient == _to) -> (__post.balances[_to] == balances
          [_to] + _value) && (__post.balances[_from] == balances[_from] - _value)
      @post (_from != _to && feeRecipient != _from && feeRecipient != _to) -> (__post.
          balances[_to] == balances[_to] + _value - _value * feeRate / feeParts) && (
          __post.balances[_from] == balances[_from] - _value) && (__post.balances[
          feeRecipient] == balances[feeRecipient] + _value * feeRate / feeParts)
     */
    function _transfer(address _from, address _to, uint256 _value) internal returns (
        uint256) {
        uint256 _fee = getFeeFor(_value);
        uint256 _principle = _value.sub(_fee);
        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(_principle);
        emit Transfer(_from, _to, _principle);
        emit Transfer(_from, feeRecipient, _fee);
        if (_fee > 0) {
            balances[feeRecipient] = balances[feeRecipient].add(_fee);
            emit FeeCollected(_from, feeRecipient, _fee);
        }

        return _principle;
    }

    // OWNER FUNCTIONALITY

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(msg.sender == owner, "onlyOwner");
        _;
    }

```

```
    /**
     * @dev Allows the current owner to begin transferring control of the contract to
         a proposedOwner
     * @param _proposedOwner The address to transfer ownership to.
     */
    /*@CTK proposeOwner
      @tag assume_completion
      @post _proposedOwner != address(0)
      @post _proposedOwner != msg.sender
      @post __post.proposedOwner == _proposedOwner
     */
    function proposeOwner(address _proposedOwner) public onlyOwner {
        require(_proposedOwner != address(0), "cannot transfer ownership to address
            zero");
        require(msg.sender != _proposedOwner, "caller already is owner");
        proposedOwner = _proposedOwner;
        emit OwnershipTransferProposed(owner, proposedOwner);
    }

    /**
     * @dev Allows the current owner or proposed owner to cancel transferring control
         of the contract to a proposedOwner
     */
    /*@CTK proposeOwner
      @tag assume_completion
      @post msg.sender == proposedOwner || msg.sender == owner
      @post proposedOwner != address(0)
      @post __post.proposedOwner == address(0)
     */
    function disregardProposeOwner() public {
        require(msg.sender == proposedOwner || msg.sender == owner, "only proposedOwner
             or owner");
        require(proposedOwner != address(0), "can only disregard a proposed owner that
            was previously set");
        address _oldProposedOwner = proposedOwner;
        proposedOwner = address(0);
        emit OwnershipTransferDisregarded(_oldProposedOwner);
    }

    /**
     * @dev Allows the proposed owner to complete transferring control of the contract
         to the proposedOwner.
     */
    /*@CTK claimOwnership
      @tag assume_completion
      @post msg.sender == proposedOwner
      @post __post.owner == proposedOwner
      @post __post.proposedOwner == address(0)
     */
    function claimOwnership() public {
        require(msg.sender == proposedOwner, "onlyProposedOwner");
        address _oldOwner = owner;
        owner = proposedOwner;
        proposedOwner = address(0);
        emit OwnershipTransferred(_oldOwner, owner);
    }

    /**
```

```
     * @dev Reclaim all PAXG at the contract address.
     * This sends the PAXG tokens that this contract add holding to the owner.
     * Note: this is not affected by freeze constraints.
     */
    /*@CTK reclaimPAXG
      @tag assume_completion
      @pre owner != this
      @post owner == msg.sender
      @post __post.balances[this] == 0
      @post __post.balances[owner] == balances[owner] + balances[this]
     */
    function reclaimPAXG() external onlyOwner {
        uint256 _balance = balances[this];
        balances[this] = 0;
        balances[owner] = balances[owner].add(_balance);
        emit Transfer(this, owner, _balance);
    }

    // PAUSABILITY FUNCTIONALITY

    /**
     * @dev Modifier to make a function callable only when the contract is not paused.
     */
    modifier whenNotPaused() {
        require(!paused, "whenNotPaused");
        _;
    }

    /**
     * @dev called by the owner to pause, triggers stopped state
     */
    /*@CTK pause
      @tag assume_completion
      @post owner == msg.sender
      @post __post.paused
     */
    function pause() public onlyOwner {
        require(!paused, "already paused");
        paused = true;
        emit Pause();
    }

    /**
     * @dev called by the owner to unpause, returns to normal state
     */
    /*@CTK unpause
      @tag assume_completion
      @post owner == msg.sender
      @post !__post.paused
     */
    function unpause() public onlyOwner {
        require(paused, "already unpaused");
        paused = false;
        emit Unpause();
    }

    // REGULATORY COMPLIANCE FUNCTIONALITY

```

```
    /**
     * @dev Sets a new regulatory compliance role address.
     * @param _newRegulatoryComplianceRole The new address allowed to freeze/unfreeze
       addresses and seize their tokens.
     */
    /*@CTK setRegulatoryComplianceRole
      @tag assume_completion
      @post msg.sender == regulatoryComplianceRole || msg.sender == owner
      @post __post.regulatoryComplianceRole == _newRegulatoryComplianceRole
     */
    function setRegulatoryComplianceRole(address _newRegulatoryComplianceRole) public
        {
        require(msg.sender == regulatoryComplianceRole || msg.sender == owner, "only
            regulatoryComplianceRole or Owner");
        emit RegulatoryComplianceRoleSet(regulatoryComplianceRole,
            _newRegulatoryComplianceRole);
        regulatoryComplianceRole = _newRegulatoryComplianceRole;
    }

    modifier onlyRegulatoryComplianceRole() {
        require(msg.sender == regulatoryComplianceRole, "onlyRegulatoryComplianceRole")
            ;
        _;
    }

    /**
     * @dev Freezes an address balance from being transferred.
     * @param _addr The new address to freeze.
     */
    /*@CTK freeze
      @tag assume_completion
      @post msg.sender == regulatoryComplianceRole
      @post !frozen[_addr]
      @post __post.frozen[_addr]
     */
    function freeze(address _addr) public onlyRegulatoryComplianceRole {
        require(!frozen[_addr], "address already frozen");
        frozen[_addr] = true;
        emit AddressFrozen(_addr);
    }

    /**
     * @dev Unfreezes an address balance allowing transfer.
     * @param _addr The new address to unfreeze.
     */
    /*@CTK unfreeze
      @tag assume_completion
      @post msg.sender == regulatoryComplianceRole
      @post frozen[_addr]
      @post !__post.frozen[_addr]
     */
    function unfreeze(address _addr) public onlyRegulatoryComplianceRole {
        require(frozen[_addr], "address already unfrozen");
        frozen[_addr] = false;
        emit AddressUnfrozen(_addr);
    }

    /**
```

```
     * @dev Wipes the balance of a frozen address, burning the tokens
     * and setting the approval to zero.
     * @param _addr The new frozen address to wipe.
     */
    /*CTK wipeFrozenAddress
      @tag assume_completion
      @post msg.sender == regulatoryComplianceRole
      @post frozen[_addr]
      @post __post.balances[_addr] == 0
      @post __post.totalSupply_ == totalSupply_ - balances[_addr]
     */
    function wipeFrozenAddress(address _addr) public onlyRegulatoryComplianceRole {
        require(frozen[_addr], "address is not frozen");
        uint256 _balance = balances[_addr];
        balances[_addr] = 0;
        totalSupply_ = totalSupply_.sub(_balance);
        emit FrozenAddressWiped(_addr);
        emit SupplyDecreased(_addr, _balance);
        emit Transfer(_addr, address(0), _balance);
    }

    /**
    * @dev Gets whether the address is currently frozen.
    * @param _addr The address to check if frozen.
    * @return A bool representing whether the given address is frozen.
    */
    /*@CTK isFrozen
      @post __return == frozen[_addr]
     */
    function isFrozen(address _addr) public view returns (bool) {
        return frozen[_addr];
    }

    // SUPPLY CONTROL FUNCTIONALITY

    /**
     * @dev Sets a new supply controller address.
     * @param _newSupplyController The address allowed to burn/mint tokens to control
         supply.
     */
    /*@CTK setSupplyController
      @tag assume_completion
      @post msg.sender == supplyController || msg.sender == owner
      @post _newSupplyController != address(0)
      @post __post.supplyController == _newSupplyController
     */
    function setSupplyController(address _newSupplyController) public {
        require(msg.sender == supplyController || msg.sender == owner, "only
            SupplyController or Owner");
        require(_newSupplyController != address(0), "cannot set supply controller to
            address zero");
        emit SupplyControllerSet(supplyController, _newSupplyController);
        supplyController = _newSupplyController;
    }

    modifier onlySupplyController() {
        require(msg.sender == supplyController, "onlySupplyController");
        _;
```

```
    }

    /**
     * @dev Increases the total supply by minting the specified number of tokens to the supply controller account.
     * @param _value The number of tokens to add.
     * @return A boolean that indicates if the operation was successful.
     */
    /*CTK increaseSupply
      @tag assume_completion
      @post msg.sender == supplyController
      @post __post.totalSupply_ == totalSupply_ + _value
      @post __post.balances[supplyController] == balances[supplyController] + _value
     */
    function increaseSupply(uint256 _value) public onlySupplyController returns (bool success) {
        totalSupply_ = totalSupply_.add(_value);
        balances[supplyController] = balances[supplyController].add(_value);
        emit SupplyIncreased(supplyController, _value);
        emit Transfer(address(0), supplyController, _value);
        return true;
    }

    /**
     * @dev Decreases the total supply by burning the specified number of tokens from the supply controller account.
     * @param _value The number of tokens to remove.
     * @return A boolean that indicates if the operation was successful.
     */
    /*CTK decreaseSupply
      @tag assume_completion
      @post msg.sender == supplyController
      @post _value <= balances[supplyController]
      @post __post.balances[supplyController] == balances[supplyController] - _value
      @post __post.totalSupply_ == totalSupply_ - _value
     */
    function decreaseSupply(uint256 _value) public onlySupplyController returns (bool success) {
        require(_value <= balances[supplyController], "not enough supply");
        balances[supplyController] = balances[supplyController].sub(_value);
        totalSupply_ = totalSupply_.sub(_value);
        emit SupplyDecreased(supplyController, _value);
        emit Transfer(supplyController, address(0), _value);
        return true;
    }

    // DELEGATED TRANSFER FUNCTIONALITY

    /**
     * @dev returns the next seq for a target address.
     * The transactor must submit nextSeqOf(transactor) in the next transaction for it to be valid.
     * Note: that the seq context is specific to this smart contract.
     * @param target The target address.
     * @return the seq.
     */
    //
    /*@CTK nextSeqOf
```

```
     @post __return == nextSeqs[target]
    */
    function nextSeqOf(address target) public view returns (uint256) {
        return nextSeqs[target];
    }

    /**
     * @dev Performs a transfer on behalf of the from address, identified by its signature on the delegatedTransfer msg.
     * Splits a signature byte array into r,s,v for convenience.
     * @param sig the signature of the delgatedTransfer msg.
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     * @param serviceFee an optional ERC20 service fee paid to the executor of betaDelegatedTransfer by the from address.
     * @param seq a sequencing number included by the from address specific to this contract to protect from replays.
     * @param deadline a block number after which the pre-signed transaction has expired.
     * @return A boolean that indicates if the operation was successful.
     */
    function betaDelegatedTransfer(
        bytes sig, address to, uint256 value, uint256 serviceFee, uint256 seq, uint256 deadline
    ) public returns (bool) {
        require(sig.length == 65, "signature should have length 65");
        bytes32 r;
        bytes32 s;
        uint8 v;
        assembly {
            r := mload(add(sig, 32))
            s := mload(add(sig, 64))
            v := byte(0, mload(add(sig, 96)))
        }
        require(_betaDelegatedTransfer(r, s, v, to, value, serviceFee, seq, deadline), "failed transfer");
        return true;
    }

    /**
     * @dev Performs a transfer on behalf of the from address, identified by its signature on the betaDelegatedTransfer msg.
     * Note: both the delegate and transactor sign in the service fees. The transactor, however,
     * has no control over the gas price, and therefore no control over the transaction time.
     * Beta prefix chosen to avoid a name clash with an emerging standard in ERC865 or elsewhere.
     * Internal to the contract - see betaDelegatedTransfer and betaDelegatedTransferBatch.
     * @param r the r signature of the delgatedTransfer msg.
     * @param s the s signature of the delgatedTransfer msg.
     * @param v the v signature of the delgatedTransfer msg.
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     * @param serviceFee an optional ERC20 service fee paid to the delegate of betaDelegatedTransfer by the from address.
```

```
 * @param seq a sequencing number included by the from address specific to this contract to protect from replays.
 * @param deadline a block number after which the pre-signed transaction has expired.
 * @return A boolean that indicates if the operation was successful.
 */
function _betaDelegatedTransfer(
    bytes32 r, bytes32 s, uint8 v, address to, uint256 value, uint256 serviceFee, uint256 seq, uint256 deadline
) internal whenNotPaused returns (bool) {
    require(betaDelegateWhitelist[msg.sender], "Beta feature only accepts whitelisted delegates");
    require(value > 0 || serviceFee > 0, "cannot transfer zero tokens with zero service fee");
    require(block.number <= deadline, "transaction expired");
    // prevent sig malleability from ecrecover()
    require(uint256(s) <= 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0, "signature incorrect");
    require(v == 27 || v == 28, "signature incorrect");

    // EIP712 scheme: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-712.md
    bytes32 hash = keccak256(abi.encodePacked(EIP191_HEADER, EIP712_DOMAIN_HASH, keccak256(abi.encodePacked(// solium-disable-line
            EIP712_DELEGATED_TRANSFER_SCHEMA_HASH, bytes32(to), value, serviceFee, seq, deadline
    ))));
    address _from = ecrecover(hash, v, r, s);

    require(_from != address(0), "error determining from address from signature");
    require(to != address(0), "cannot use address zero");
    require(!frozen[to] && !frozen[_from] && !frozen[msg.sender], "address frozen");
    require(value.add(serviceFee) <= balances[_from], "insufficient funds or bad signature");
    require(nextSeqs[_from] == seq, "incorrect seq");

    nextSeqs[_from] = nextSeqs[_from].add(1);

    uint256 _principle = _transfer(_from, to, value);

    if (serviceFee != 0) {
        balances[_from] = balances[_from].sub(serviceFee);
        balances[msg.sender] = balances[msg.sender].add(serviceFee);
        emit Transfer(_from, msg.sender, serviceFee);
    }

    emit BetaDelegatedTransfer(_from, to, _principle, seq, serviceFee);
    return true;
}

/**
 * @dev Performs an atomic batch of transfers on behalf of the from addresses, identified by their signatures.
 * Lack of nested array support in arguments requires all arguments to be passed as equal size arrays where
 * delegated transfer number i is the combination of all arguments at index i
 * @param r the r signatures of the delgatedTransfer msg.
```

```
     * @param s the s signatures of the delgatedTransfer msg.
     * @param v the v signatures of the delgatedTransfer msg.
     * @param to The addresses to transfer to.
     * @param value The amounts to be transferred.
     * @param serviceFee optional ERC20 service fees paid to the delegate of
         betaDelegatedTransfer by the from address.
     * @param seq sequencing numbers included by the from address specific to this
         contract to protect from replays.
     * @param deadline block numbers after which the pre-signed transactions have
         expired.
     * @return A boolean that indicates if the operation was successful.
     */
    function betaDelegatedTransferBatch(
        bytes32[] r, bytes32[] s, uint8[] v, address[] to, uint256[] value, uint256[]
            serviceFee, uint256[] seq, uint256[] deadline
    ) public returns (bool) {
        require(r.length == s.length && r.length == v.length && r.length == to.length
            && r.length == value.length, "length mismatch");
        require(r.length == serviceFee.length && r.length == seq.length && r.length ==
            deadline.length, "length mismatch");

        for (uint i = 0; i < r.length; i++) {
            require(
                _betaDelegatedTransfer(r[i], s[i], v[i], to[i], value[i], serviceFee[i
                    ], seq[i], deadline[i]),
                "failed transfer"
            );
        }
        return true;
    }

    /**
    * @dev Gets whether the address is currently whitelisted for betaDelegateTransfer.
    * @param _addr The address to check if whitelisted.
    * @return A bool representing whether the given address is whitelisted.
    */
    /*@CTK isWhitelistedBetaDelegate
      @post __return == betaDelegateWhitelist[_addr]
     */
    function isWhitelistedBetaDelegate(address _addr) public view returns (bool) {
        return betaDelegateWhitelist[_addr];
    }

    /**
     * @dev Sets a new betaDelegate whitelister.
     * @param _newWhitelister The address allowed to whitelist betaDelegates.
     */
    /*@CTK setBetaDelegateWhitelister
      @tag assume_completion
      @post msg.sender == betaDelegateWhitelister || msg.sender == owner
      @post __post.betaDelegateWhitelister == _newWhitelister
     */
    function setBetaDelegateWhitelister(address _newWhitelister) public {
        require(msg.sender == betaDelegateWhitelister || msg.sender == owner, "only
            Whitelister or Owner");
        betaDelegateWhitelister = _newWhitelister;
        emit BetaDelegateWhitelisterSet(betaDelegateWhitelister, _newWhitelister);
    }
```

```
    modifier onlyBetaDelegateWhitelister() {
        require(msg.sender == betaDelegateWhitelister, "onlyBetaDelegateWhitelister");
        _;
    }

    /**
     * @dev Whitelists an address to allow calling BetaDelegatedTransfer.
     * @param _addr The new address to whitelist.
     */
    /*@CTK whitelistBetaDelegate
      @tag assume_completion
      @post msg.sender == betaDelegateWhitelister
      @post !betaDelegateWhitelist[_addr]
      @post __post.betaDelegateWhitelist[_addr]
     */
    function whitelistBetaDelegate(address _addr) public onlyBetaDelegateWhitelister {
        require(!betaDelegateWhitelist[_addr], "delegate already whitelisted");
        betaDelegateWhitelist[_addr] = true;
        emit BetaDelegateWhitelisted(_addr);
    }

    /**
     * @dev Unwhitelists an address to disallow calling BetaDelegatedTransfer.
     * @param _addr The new address to whitelist.
     */
    /*@CTK unwhitelistBetaDelegate
      @tag assume_completion
      @post msg.sender == betaDelegateWhitelister
      @post betaDelegateWhitelist[_addr]
      @post !__post.betaDelegateWhitelist[_addr]
     */
    function unwhitelistBetaDelegate(address _addr) public onlyBetaDelegateWhitelister
        {
        require(betaDelegateWhitelist[_addr], "delegate not whitelisted");
        betaDelegateWhitelist[_addr] = false;
        emit BetaDelegateUnwhitelisted(_addr);
    }

    // FEE CONTROLLER FUNCTIONALITY

    /**
     * @dev Sets a new fee controller address.
     * @param _newFeeController The address allowed to set the fee rate and the fee
        recipient.
     */
    /*@CTK setFeeController
      @tag assume_completion
      @post msg.sender == feeController || msg.sender == owner
      @post _newFeeController != address(0)
      @post __post.feeController == _newFeeController
     */
    function setFeeController(address _newFeeController) public {
        require(msg.sender == feeController || msg.sender == owner, "only FeeController
            or Owner");
        require(_newFeeController != address(0), "cannot set fee controller to address
            zero");
        address _oldFeeController = feeController;
```

```
        feeController = _newFeeController;
        emit FeeControllerSet(_oldFeeController, feeController);
    }

    modifier onlyFeeController() {
        require(msg.sender == feeController, "only FeeController");
        _;
    }

    /**
     * @dev Sets a new fee recipient address.
     * @param _newFeeRecipient The address allowed to collect transfer fees for
       transfers.
     */
    /*@CTK setFeeRecipient
      @tag assume_completion
      @post msg.sender == feeController
      @post _newFeeRecipient != address(0)
      @post __post.feeRecipient == _newFeeRecipient
     */
    function setFeeRecipient(address _newFeeRecipient) public onlyFeeController {
        require(_newFeeRecipient != address(0), "cannot set fee recipient to address
            zero");
        address _oldFeeRecipient = feeRecipient;
        feeRecipient = _newFeeRecipient;
        emit FeeRecipientSet(_oldFeeRecipient, feeRecipient);
    }

    /**
     * @dev Sets a new fee rate.
     * @param _newFeeRate The new fee rate to collect as transfer fees for transfers.
     */
    /*@CTK setFeeRate
      @tag assume_completion
      @post msg.sender == feeController
      @post _newFeeRate <= feeParts
      @post __post.feeRate == _newFeeRate
     */
    function setFeeRate(uint256 _newFeeRate) public onlyFeeController {
        require(_newFeeRate <= feeParts, "cannot set fee rate above 100%");
        uint256 _oldFeeRate = feeRate;
        feeRate = _newFeeRate;
        emit FeeRateSet(_oldFeeRate, feeRate);
    }

    /**
    * @dev Gets a fee for a given value
    * ex: given feeRate = 200 and feeParts = 1,000,000 then getFeeFor(10000) = 2
    * @param _value The amount to get the fee for.
    */
    /*@CTK getFeeFor
      @tag assume_completion
      @post (feeRate == 0) -> (__return == 0)
      @post (feeRate != 0) -> (__return == _value * feeRate / feeParts)
     */
    function getFeeFor(uint256 _value) public view returns (uint256) {
        if (feeRate == 0) {
            return 0;
```

```
        }

        return _value.mul(feeRate).div(feeParts);
    }
}
```