0.7

Idex.io Process Quality Review

Score: 76%

Overview

This is a Process Quality Review of Idex completed on May 10, 2021. It was performed using the Process Review process (version 0.7) and is documented here. The review was performed by Lucas of DeFiSafety. Check out our Telegram.

The final score of the review is 76%, a pass. The breakdown of the scoring is in Scoring Appendix. For our purposes, a pass is 70%.

Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**
- **Here are the admin controls and strategies**

Disclaimer

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

Chain

This section indicates the blockchain used by this protocol.

> ✓ **Chain: Ethereum/Binance**

Guidance:
Ethereum
Binance

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is here. This review will answer the questions;

1) Are the executing code addresses readily available? (%)
2) Is the code actively being used? (%)
3) Is there a public software repository? (Y/N)
4) Is there a development history visible? (%)
5) Is the team public (not anonymous)? (Y/N)

1) Are the executing code addresses readily available? (%)

> ⓘ Answer: 70%

The address was clearly labelled on their documentation, but was not easy to find.

They are available at website https://docs.idex.io/#url-amp-contract-addresses as indicated in the Appendix.

Guidance:
100% Clearly labelled and on website, docs or repo, quick to find
70% Clearly labelled and on website, docs or repo but takes a bit of looking
40% Addresses in mainnet.json, in discord or sub graph, etc
20% Address found but labelling not clear or easy to find
0% Executing addresses could not be found

How to improve this score

Make the Ethereum addresses of the smart contract utilized by your application available on either your website or your GitHub (in the README for instance). Ensure the addresses is up to date. This is a very important question wrt to the final score.

2) Is the code actively being used? (%)

> Answer: 100%

Activity is 80 transactions a day on contract *exchange.sol*, as indicated in the Appendix.

Percentage Score Guidance

100%	More than 10 transactions a day
70%	More than 10 transactions a week
40%	More than 10 transactions a month
10%	Less than 10 transactions a month
0%	No activity

3) Is there a public software repository? (Y/N)

> Answer: Yes

GitHub: https://github.com/idexio/idex-contracts

Is there a public software repository with the code at a minimum, but normally test and scripts also (Y/N). Even if the repo was created just to hold the files and has just 1 transaction, it gets a Yes. For teams with private repos, this answer is No.

4) Is there a development history visible? (%)

> Answer: 70%

With 92 commits and 9 branches this is a semi-healthy repository.

This checks if the software repository demonstrates a strong steady history. This is normally demonstrated by commits, branches and releases in a software repository. A healthy history demonstrates a history of more than a month (at a minimum).

Guidance:

100%	Any one of 100+ commits, 10+branches
70%	Any one of 70+ commits, 7+branches

50% Any one of 50+ commits, 5+branches
30% Any one of 30+ commits, 3+branches
0% Less than 2 branches or less than 10 commits

How to improve this score

Continue to test and perform other verification activities after deployment, including routine maintenance updating to new releases of testing and deployment tools. A public development history indicates clearly to the public the level of continued investment and activity by the developers on the application. This gives a level of security and faith in the application.

5) Is the team public (not anonymous)? (Y/N)

> Answer: Yes

Team names on Github; https://github.com/idexio/idex-contracts

For a yes in this question the real names of some team members must be public on the website or other documentation. If the team is anonymous and then this question is a No.

Documentation

This section looks at the software documentation. The document explaining these questions is here.

Required questions are;

6) Is there a whitepaper? (Y/N)
7) Are the basic software functions documented? (Y/N)
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)
10) Is it possible to trace from software documentation to the implementation in
code (%)

6) Is there a whitepaper? (Y/N)

> Answer: Yes

Location: https://idex.io/document/IDEX-2-0-Whitepaper-2019-10-31.pdf

7) Are the basic software functions documented? (Y/N)

> Answer: Yes

There is robust API documentation available.

How to improve this score

Write the document based on the deployed code. For guidance, refer to the SecurEth System Description Document.

8) Does the software function documentation fully (100%) cover the deployed contracts? (%)

Answer: 80%

There is robust API documentation of any API contracts.

Guidance:

100% All contracts and functions documented
80% Only the major functions documented
79-1% Estimate of the level of software documentation
0% No software documentation

How to improve this score

This score can improve by adding content to the requirements document such that it comprehensively covers the requirements. For guidance, refer to the SecurEth System Description Document . Using tools that aid traceability detection will help.

9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)

Answer: 51%

Code examples are in the Appendix. As per the SLOC, there is 51% commenting to code (CtC).

The Comments to Code (CtC) ratio is the primary metric for this score.

Guidance:
100% CtC > 100 Useful comments consistently on all code
90-70% CtC > 70 Useful comment on most code
60-20% CtC > 20 Some useful commenting
0% CtC < 20 No useful commenting

How to improve this score

This score can improve by adding comments to the deployed code such that it comprehensively covers the code. For guidance, refer to the SecurEth Software Requirements.

10) Is it possible to trace from software documentation to the implementation in code (%)

> ⚠ Answer: 0%

While the robust API code does implicitly document many external functions, the actual code is never mentioned so no traceability

Guidance:
100% Clear explicit traceability between code and documentation at a requirement level for all code
60% Clear association between code and documents via non explicit traceability
40% Documentation lists all the functions and describes their functions
0% No connection between documentation and code

How to improve this score

This score can improve by adding traceability from requirements to code such that it is clear where each requirement is coded. For reference, check the SecurEth guidelines on traceability.

Testing

This section looks at the software testing available. It is explained in this document. This section answers the following questions;

11) Full test suite (Covers all the deployed code) (%)
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)
13) Scripts and instructions to run the tests (Y/N)
14) Report of the results (%)
15) Formal Verification test done (%)
16) Stress Testing environment (%)

11) Is there a Full test suite? (%)

> ✓ Answer: 100%

With a TtC of 278%, there is clearly a robust set of tests.

This score is guided by the Test to Code ratio (TtC). Generally a good test to code ratio is over 100%. However the reviewers best judgement is the final deciding factor.

Guidance:
100% TtC > 120% Both unit and system test visible
80% TtC > 80% Both unit and system test visible

40% TtC < 80% Some tests visible
0% No tests obvious

How to improve this score

This score can improve by adding tests to fully cover the code. Document what is covered by traceability or test results in the software repository.

12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)

> Answer: 100%

Their Audit by Quantstamp points to a 100% code coverage.

Guidance:
100% Documented full coverage
99-51% Value of test coverage from documented results
50% No indication of code coverage but clearly there is a reasonably complete set of tests
30% Some tests evident but not complete
0% No test for coverage seen

How to improve this score

This score can improve by adding tests achieving full code coverage. A clear report and scripts in the software repository will guarantee a high score.

13) Scripts and instructions to run the tests (Y/N)

> Answer: Yes

Location: https://github.com/idexio/idex-contracts

How to improve this score

Add the scripts to the repository and ensure they work. Ask an outsider to create the environment and run the tests. Improve the scripts and docs based on their feedback.

14) Report of the results (%)

> Answer: 70%

Code coverage results in the GitHub coveralls results.

Guidance:
100% Detailed test report as described below
70% GitHub Code coverage report visible
0% No test report evident

How to improve this score

Add a report with the results. The test scripts should generate the report or elements of it.

15) Formal Verification test done (%)

⚠ Answer: 0%

16) Stress Testing environment (%)

✓ Answer: 100%

Their contracts have been clearly well-tested on the RinkBy testnet.

Security

This section looks at the 3rd party software audits done. It is explained in this document. This section answers the following questions;

17) Did 3rd Party audits take place? (%)
18) Is the bounty value acceptably high?

17) Did 3rd Party audits take place? (%)

✓ Answer: 90%

Quantstamp preformed an audit on IDEX V2 on July 21st 2020.

IDEX V2 was released October 16th.

Guidance:
100% Multiple Audits performed before deployment and results public and implemented or not required
90% Single audit performed before deployment and results public and implemented or not required

70% Audit(s) performed after deployment and no changes required. Audit report is public
20% No audit performed
0% Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, question

18) Is the bounty value acceptably high (%)

> ⚠ Answer: 0%

There is no apparent bug bounty program.

Guidance:

100% Bounty is 10% TVL or at least $1M AND active program (see below)
90% Bounty is 5% TVL or at least 500k AND active program
80% Bounty is 5% TVL or at least 500k
70% Bounty is 100k or over AND active program
50% Bounty is 100k or over
40% Bounty is 50k or over
20% Bug bounty program bounty is less than 50k
0% No bug bounty program offered

Active program means a third party actively driving hackers to the site. Inactive program would be static mention on the docs.

Access Controls

This section covers the documentation of special access controls for a DeFi protocol. The admin access controls are the contracts that allow updating contracts or coefficients in the protocol. Since these contracts can allow the protocol admins to "change the rules", complete disclosure of capabilities is vital for user's transparency. It is explained in this document. The questions this section asks are as follow;

19) Can a user clearly and quickly find the status of the admin controls?
20) Is the information clear and complete?
2`) Is the information in non-technical terms that pertain to the investments?
22) Is there Pause Control documentation including records of tests?

19) Can a user clearly and quickly find the status of the admin controls (%)

> ⓘ Answer: 70%

Idex.io has a Governance.md file that contains information about admin controls.

Guidance:
100% Clearly labelled and on website, docs or repo, quick to find
70% Clearly labelled and on website, docs or repo but takes a bit of looking
40% Access control docs in multiple places and not well labelled
20% Access control docs in multiple places and not labelled
0% Admin Control information could not be found

20) Is the information clear and complete (%)

> Answer: 90%

The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles)

All contracts are clearly labelled as upgradeable (or not)

The capabilities for change in the contracts are described

Guidance:
All the contracts are immutable -- 100% OR

All contracts are clearly labelled as upgradeable (or not) -- 30% AND
The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
The capabilities for change in the contracts are described -- 30%

How to improve this score

Create a document that covers the items described above. An example is enclosed.

21) Is the information in non-technical terms that pertain to the investments (%)

> Answer: 90%

Guidance:
100% All the contracts are immutable
90% Description relates to investments safety and updates in clear, complete non-software l language
30% Description all in software specific language
0% No admin control information could not be found

How to improve this score

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

22) Is there Pause Control documentation including records of tests (%)

⚠ Answer: 0%

Guidance:

100% All the contracts are immutable or no pause control needed and this is explained OR

100% Pause control(s) are clearly documented and there is records of at least one test within 3 months

80% Pause control(s) explained clearly but no evidence of regular tests

40% Pause controls mentioned with no detail on capability or tests

0% Pause control not documented or explained

How to improve this score

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

I was coaxed to go to EthDenver 2018 and there I started SecuEth.org with Bryant and Roman. We created guidelines on good processes for blockchain code development. We got EthFoundation funding to assist in their development.

Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix

	Total	idex.io	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		198.55
Code and Team			**76%**
1) Are the executing code addresses readily available? (%)	**20**	70%	14
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**5**	y	5
4) Is there a development history visible? (%)	**5**	70%	3.5
5) Is the team public (not anonymous)? (Y/N)	**15**	Y	15

Code Documentation			
6) Is there a whitepaper? (Y/N)	**5**	y	5
7) Are the basic software functions documented? (Y/N)	**10**	Y	10
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)	**15**	80%	12
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	51%	2.55
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	0%	0
Testing			
11) Full test suite (Covers all the deployed code) (%)	**20**	100%	20
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	100%	5
13) Scripts and instructions to run the tests? (Y/N)	**5**	Y	5
14) Report of the results (%)	**10**	70%	7
15) Formal Verification test done (%)	**5**	0%	0
16) Stress Testing environment (%)	**5**	100%	5
Security			
17) Did 3rd Party audits take place? (%)	**70**	90%	63
18) Is the bug bounty acceptable high? (%)	**10**	0%	0
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	70%	3.5
20) Is the information clear and complete	**10**	90%	9
21) Is the information in non-technical terms	**10**	90%	9
22) Is there Pause Control documentation including records of tests	**10**	0%	0
Section Scoring			
Code and Team	50	85%	
Documentation	45	66%	
Testing	50	84%	
Security	80	79%	
Access Controls	35	61%	

Executing Code Appendix

REST API Interaction

URL & Contract Addresses

Each blockchain supported by IDEX has a dedicated host for REST API requests. **Requests to one chain's endpoint only returns data specific to that chain.** For example, calling the Get Markets endpoint for Binance Smart Chain only returns the markets available on BSC.

Ethereum

REST API: https://api-eth.idex.io/
Exchange Contract: 0xA36972E347E538E6C7Afb9f44FB10DDa7BBa9BA2

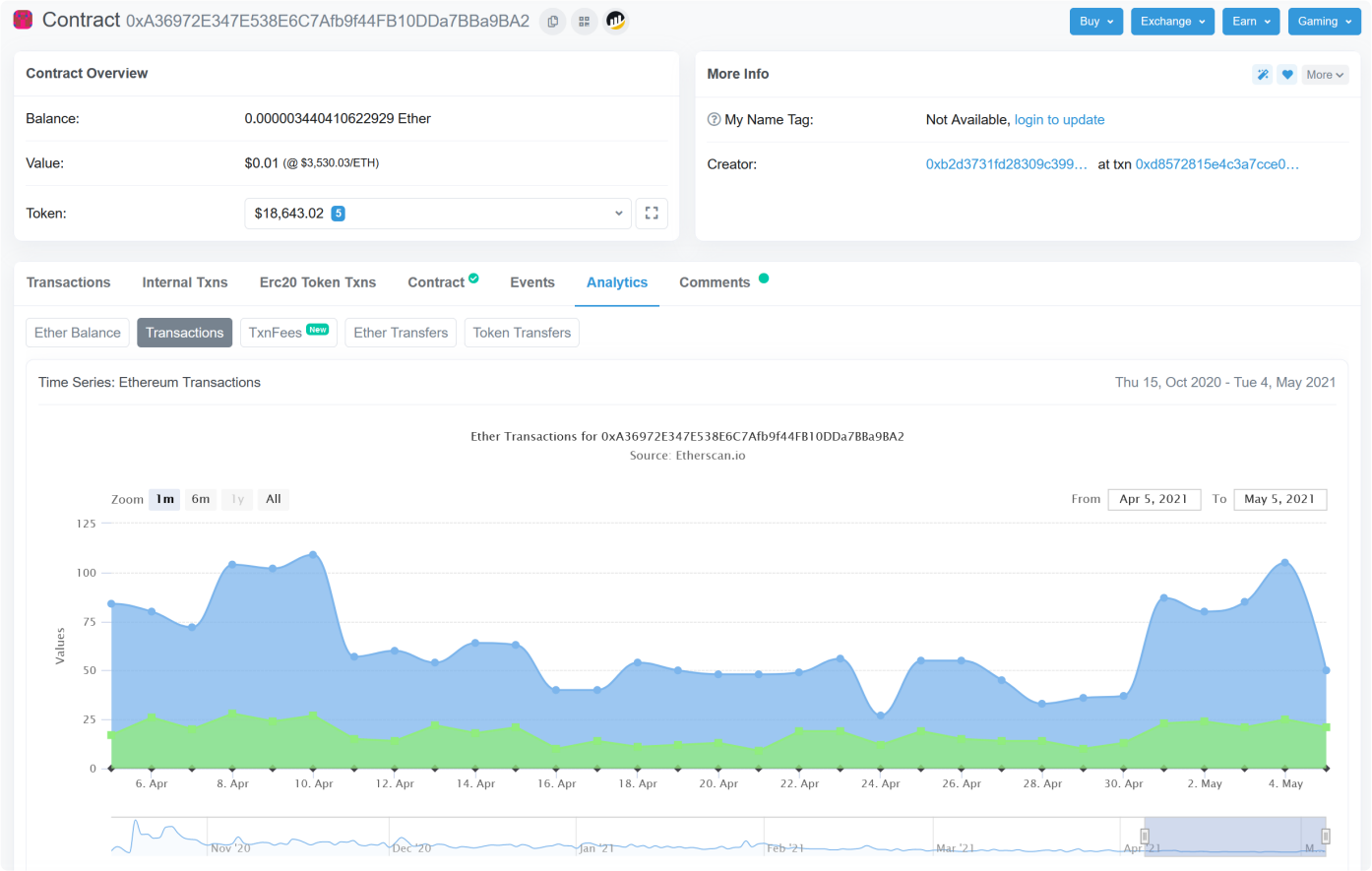
The previous Ethereum REST API endpoint, https://api.idex.io/, is deprecated but will continue to work for Ethereum-related requests.

Binance Smart Chain

REST API: https://api-bsc.idex.io/
Exchange Contract: 0x8C788aA08A98002413F6350C29c017aefb2c08C7

IDEX's data centers are in the AWS Europe (Ireland) `eu-west-1` region.

Code Used Appendix



Example Code Appendix

```
// SPDX-License-Identifier: LGPL-3.0-only

pragma solidity 0.6.8;
pragma experimental ABIEncoderV2;

import { Address } from '@openzeppelin/contracts/utils/Address.sol';
import {
  SafeMath as SafeMath256
} from '@openzeppelin/contracts/math/SafeMath.sol';

import { ICustodian } from './libraries/Interfaces.sol';
import { Owned } from './Owned.sol';


contract Governance is Owned {
  using SafeMath256 for uint256;

  /**
   * @notice Emitted when admin initiates upgrade of `Exchange` contract address on `Custoc
   * `initiateExchangeUpgrade`
   */
  event ExchangeUpgradeInitiated(
    address oldExchange,

```

```
    address newExchange,
    uint256 blockThreshold
  );
  /**
   * @notice Emitted when admin cancels previously started `Exchange` upgrade with `cancel
   */
  event ExchangeUpgradeCanceled(address oldExchange, address newExchange);
  /**
   * @notice Emitted when admin finalizes `Exchange` upgrade via `finalizeExchangeUpgrade`
   */
  event ExchangeUpgradeFinalized(address oldExchange, address newExchange);
  /**
   * @notice Emitted when admin initiates upgrade of `Governance` contract address on `Cus
   * `initiateGovernanceUpgrade`
   */
  event GovernanceUpgradeInitiated(
    address oldGovernance,
    address newGovernance,
    uint256 blockThreshold
  );
  /**
   * @notice Emitted when admin cancels previously started `Governance` upgrade with `canc
   */
  event GovernanceUpgradeCanceled(address oldGovernance, address newGovernance);
  /**
   * @notice Emitted when admin finalizes `Governance` upgrade via `finalizeGovernanceUpgr
   * this contract and rendering it non-functioning
   */
  event GovernanceUpgradeFinalized(
    address oldGovernance,
    address newGovernance
  );

  // Internally used structs //

  struct ContractUpgrade {
    bool exists;
    address newContract;
    uint256 blockThreshold;
  }

  // Storage //

  uint256 immutable _blockDelay;
  ICustodian _custodian;
  ContractUpgrade _currentExchangeUpgrade;
  ContractUpgrade _currentGovernanceUpgrade;

  /**
   * @notice Instantiate a new `Governance` contract
   *
   * @dev Sets `owner` and `admin` to `msg.sender`. Sets the values for `_blockDelay` gove
   * and `Governance` upgrades. This value is immutable, and cannot be changed after const
```

```
77
     *
78   * @param blockDelay The minimum number of blocks that must be mined after initiating an
79   * or `Governance` upgrade before the upgrade may be finalized
80   */
81  constructor(uint256 blockDelay) public Owned() {
82    _blockDelay = blockDelay;
83  }
84
85  /**
86   * @notice Sets the address of the `Custodian` contract. The `Custodian` accepts `Exchang
87   * `Governance` addresses in its constructor, after which they can only be changed by the
88   * `Governance` contract itself. Therefore the `Custodian` must be deployed last and its
89   * set here on an existing `Governance` contract. This value is immutable once set and ca
90   * changed again
91   *
92   * @param newCustodian The address of the `Custodian` contract deployed against this `Gov
93   * contract's address
94   */
95  function setCustodian(ICustodian newCustodian) external onlyAdmin {
96    require(_custodian == ICustodian(0x0), 'Custodian can only be set once');
97    require(Address.isContract(address(newCustodian)), 'Invalid address');
98
99    _custodian = newCustodian;
100 }
101
102 // Exchange upgrade //
103
104 /**
105  * @notice Initiates `Exchange` contract upgrade proccess on `Custodian`. Once `blockDela
106  * the process can be finalized with `finalizeExchangeUpgrade`
107  *
108  * @param newExchange The address of the new `Exchange` contract
109  */
110 function initiateExchangeUpgrade(address newExchange) external onlyAdmin {
111   require(Address.isContract(address(newExchange)), 'Invalid address');
112   require(
113     newExchange != _custodian.loadExchange(),
114     'Must be different from current Exchange'
115   );
116   require(
117     !_currentExchangeUpgrade.exists,
118     'Exchange upgrade already in progress'
119   );
120
121   _currentExchangeUpgrade = ContractUpgrade(
122     true,
123     newExchange,
124     block.number.add(_blockDelay)
125   );
126
127   emit ExchangeUpgradeInitiated(
128     _custodian.loadExchange(),
129     newExchange
```

```
      newExchange,
      _currentExchangeUpgrade.blockThreshold
    );
  }

  /**
   * @notice Cancels an in-flight `Exchange` contract upgrade that has not yet been finali
   */
  function cancelExchangeUpgrade() external onlyAdmin {
    require(_currentExchangeUpgrade.exists, 'No Exchange upgrade in progress');

    address newExchange = _currentExchangeUpgrade.newContract;
    delete _currentExchangeUpgrade;

    emit ExchangeUpgradeCanceled(_custodian.loadExchange(), newExchange);
  }

  /**
   * @notice Finalizes the `Exchange` contract upgrade by changing the contract address on
   * contract with `setExchange`. The number of blocks specified by `_blockDelay` must have
   * `initiateExchangeUpgrade`
   *
   * @param newExchange The address of the new `Exchange` contract. Must equal the address
   * `initiateExchangeUpgrade`
   */
  function finalizeExchangeUpgrade(address newExchange) external onlyAdmin {
    require(_currentExchangeUpgrade.exists, 'No Exchange upgrade in progress');
    require(
      _currentExchangeUpgrade.newContract == newExchange,
      'Address mismatch'
    );
    require(
      block.number >= _currentExchangeUpgrade.blockThreshold,
      'Block threshold not yet reached'
    );

    address oldExchange = _custodian.loadExchange();
    _custodian.setExchange(newExchange);
    delete _currentExchangeUpgrade;

    emit ExchangeUpgradeFinalized(oldExchange, newExchange);
  }

  // Governance upgrade //

  /**
   * @notice Initiates `Governance` contract upgrade proccess on `Custodian`. Once `blockDe
   * the process can be finalized with `finalizeGovernanceUpgrade`
   *
   * @param newGovernance The address of the new `Governan
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	12	2617	304	782	1531	110

Comments to Code 782/1531 = 51%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	22	4963	629	66	4268	160

Tests to Code 4268/1531 = 278%