# 0.7

## Tokemak Process Quality Review

Score: 56%

## Overview

This is a Tokemak Process Quality Review completed on October 14th 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Nic of DeFiSafety. Check out our Telegram.

The final score of the review is **56%**, a **FAIL**. 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 blockchains used by this protocol. This report covers all of the blockchains upon which the protocol is deployed.

> **Chain:** Ethereum

**Guidance:**

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra
Celo
Arbitrum
Solana

---

## 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 following 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:** 100%

They are available at website https://tokemak.gitbook.io/tokemak/protocol-information/contract-interactions, 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 labeling not clear or easy to find |
| 0% | Executing addresses could not be found |

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

> ✓ **Answer:** 100%

Activity is over 10 transactions a day on contract *Pool.sol*, as indicated in the Appendix.

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/Tokemak/tokemak-smart-contracts-public.

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

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

> ⚠ **Answer:** 0%

With 13 commits and 1 branch, Tokemak's main contract repository is underdevelopped.

This metric 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 30 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

**Location:** https://github.com/Tokemak/tokemak-smart-contracts-public/graphs/contributors.

For a **"Yes"** in this question, the real names of some team members must be public on the website or other documentation (LinkedIn, etc). If the team is anonymous, 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://tokemak.gitbook.io/tokemak/.

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

> ⚠ **Answer:** No

There are no software functions documented in the Tokemak documentation.

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:** 0%

There are no software functions documented in the Tokemak documentation.

**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 be improved by adding content to the software functions 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:** 0%

Code examples are in the Appendix. As per the SLOC, there is 7% 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%

As there are no software functions documented in the Tokemak documentation, it is impossible for us to evaluate the degree of traceability as to their implementation in the protocol's source code.

**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 documentation to code such that it is clear where each outlined function is coded in the source code. 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:** 0%

There is no test suite in the Tokemak GitHub repository.

**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 improved 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:** 85%

Tokemak has an average of 85% code coverage in their Quantstamp audit.

**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 improved by adding tests that achieve 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:** No

There are no scripts or test instructions in the Tokemak GitHub repositories.

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:** 0%

There is no Tokemak test result report in their GitHub repositories.

**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%

Tokemak has not yet undergone a Formal Verification test.

**16) Stress Testing environment (%)**

> ⚠ **Answer:** 0%

There is no evidence of Tokemak's testnet smart contract usage in any of their documentation.

---

## 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:** 100%

Tokemak has had two audits deployed before mainnet launches, fixes have been implemented, and the

public reports can be found at https://tokemak.gitbook.io/tokemak/protocol-information/network-security.

**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

50% Audit(s) performed after deployment and changes needed but not implemented
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, (where question 1 is 0%)

Deduct 25% if code is in a private repo and no note from auditors that audit is applicable to deployed code

**18) Is the bounty value acceptably high (%)**

> **Answer:** 100%

Tokemak's Bug Bounty program rewards participating users with up to 1.5m for the most critical of finds. Program can be found at https://immunefi.com/bounty/tokemak/.

**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
60% Bounty is 100k or over
50% Bounty is 50k or over AND active program
40% Bounty is 50k or over
20% Bug bounty program bounty is less than 50k
0% No bug bounty program offered

An active program means that a third party (such as Immunefi) is actively driving hackers to the site. An inactive program would be static mentions 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?
21) 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 access controls (%)**

> ⓘ **Answer:** 70%

Although an official governance model has yet to be released on Tokemak's end, you can already find some decent information about DAO capabilities at https://medium.com/tokemak/tokenomics-4b3857badc73, https://tokemak.gitbook.io/tokemak/governance-+-treasury/untitled, as well as https://tokemak.gitbook.io/tokemak/toke/toke-voting-and-orbitdb.

**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:** 50%

a) All contracts are clearly labelled as upgradeable - 10% - only the Reactor contracts are clearly labelled as upgradeable for now.

b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) - 30% - as clear voting roles are already established, and there are MultiSig smart contracts detailed at https://github.com/Tokemak/tokemak-smart-contracts-public.

c) The capabilities for change in the contracts are described - 10% - as only the Reactor contracts have clear and established capabilities for change, and there is additional information as to potential change implementation at the bottom of https://medium.com/tokemak/tokenomics-4b3857badc73.

**Guidance:**

All the contracts are immutable -- 100% OR

a) All contracts are clearly labelled as upgradeable (or not) -- 30% AND
b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
c) 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:** 30%

All Tokemak access control documentation is currently in technical software language, as no true user investment safety information is currently available.

**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%

There is currently no Pause Control or similar functions detailed in the Tokemak documenation.

**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 | Tokemak | |
|---|---|---|---|
| **PQ Audit Scoring Matrix (v0.7)** | **Points** | **Answer** | **Points** |
| Total | **260** | | 145.75 |
| **Code and Team** | | | **56%** |
| 1) Are the executing code addresses readily available? (%) | **20** | 100% | 20 |
| 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** | 0% | 0 |
| 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** | n | 0 |
| 8) Does the software function documentation fully (100%) co | **15** | 0% | 0 |
| 9) Are there sufficiently detailed comments for all functions w | **5** | 0% | 0 |
| 10) Is it possible to trace from software documentation to the | **10** | 0% | 0 |
| **Testing** | | | |
| 11) Full test suite (Covers all the deployed code) (%) | **20** | 0% | 0 |
| 12) Code coverage (Covers all the deployed lines of code, or ex | **5** | 85% | 4.25 |
| 13) Scripts and instructions to run the tests? (Y/N) | **5** | n | 0 |
| 14) Report of the results (%) | **10** | 0% | 0 |
| 15) Formal Verification test done (%) | **5** | 0% | 0 |
| 16) Stress Testing environment (%) | **5** | 0% | 0 |
| **Security** | | | |
| 17) Did 3rd Party audits take place? (%) | **70** | 100% | 70 |
| 18) Is the bug bounty acceptable high? (%) | **10** | 100% | 10 |
| **Access Controls** | | | |
| 19) Can a user clearly and quickly find the status of the admin | **5** | 70% | 3.5 |
| 20) Is the information clear and complete | **10** | 50% | 5 |

| | | | |
|---|---|---|---|
| 20) Is the information clear and complete | 10 | 30% | 3 |
| 21) Is the information in non-technical terms | 10 | 30% | 3 |
| 22) Is there Pause Control documentation including records of | 10 | 0% | 0 |
| | | | |
| **Section Scoring** | | | |
| Code and Team | 50 | 90% | |
| Documentation | 45 | 11% | |
| Testing | 50 | 9% | |
| Security | 80 | 100% | |
| Access Controls | 35 | 33% | |

## Executing Code Appendix

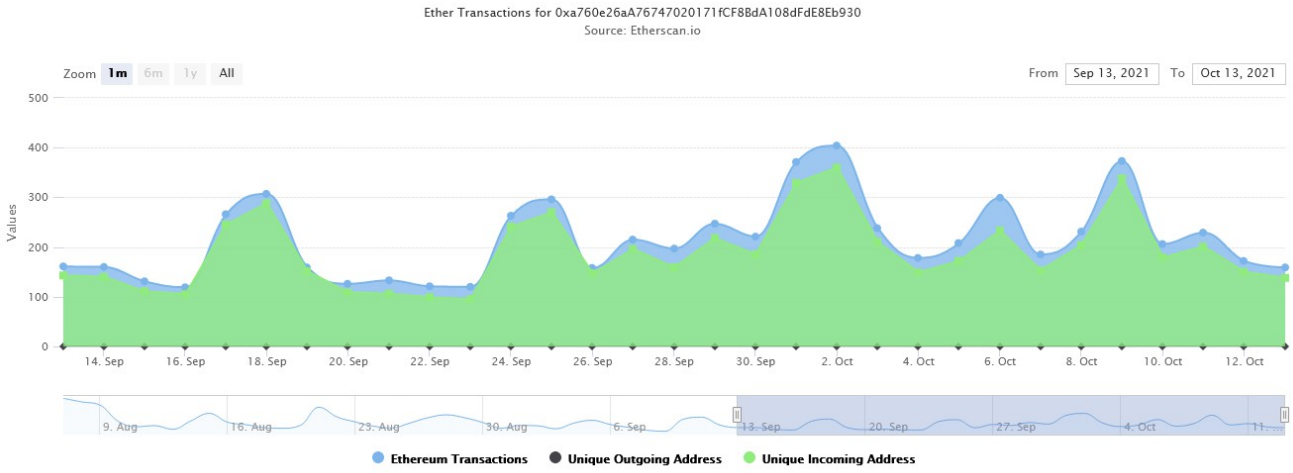
*Deployer Address* - `0x9e0bcE7ec474B481492610eB9dd5D69EB03718D5`

*TOKE Contract* - `0x2e9d63788249371f1DFC918a52f8d799F4a38C94`

*DeGenesis ("DeFi") Contract* - `0xc803737D3E12CC4034Dde0B2457684322100Ac38`

*Manager Contract* - `0xA86e412109f77c45a3BC1c5870b880492Fb86A14`

## Code Used Appendix




## Example Code Appendix

```
contract Pool is ILiquidityPool, Initializable, ERC20, Ownable, Pausable {
    using SafeMath for uint256;
    using SafeERC20 for ERC20;

    ERC20 public override underlyer; // Underlying ERC20 token

    IManager public manager;

```

```
    // implied: deployableLiquidity = underlyer.balanceOf(this) - withheldLiquidity
    uint256 public override withheldLiquidity;

    // fAsset holder -> WithdrawalInfo
    mapping(address => WithdrawalInfo) public override requestedWithdrawals;

    // NonReentrant
    bool private _entered;
    bool public _eventSend;
    Destinations public destinations;

    modifier nonReentrant() {
        require(!_entered, "ReentrancyGuard: reentrant call");
        _entered = true;
        _;
        _entered = false;
    }

    modifier onEventSend() {
        if(_eventSend) {
            _;
        }
    }

    function initialize(
        ERC20 _underlyer,
        IManager _manager,
        string memory _name,
        string memory _symbol
    ) public initializer {
        require(address(_underlyer) != address(0), "ZERO_ADDRESS");
        require(address(_manager) != address(0), "ZERO_ADDRESS");

        __Context_init_unchained();
        __Ownable_init_unchained();
        __Pausable_init_unchained();
        __ERC20_init_unchained(_name, _symbol);

        underlyer = _underlyer;
        manager = _manager;
    }

    ///@notice Gets decimals of underlyer so that tAsset decimals will match
    function decimals() public view override returns (uint8) {
        return underlyer.decimals();
    }

    function deposit(uint256 amount) external override whenNotPaused {
        _deposit(msg.sender, msg.sender, amount);
    }

    function depositFor(address account, uint256 amount) external override whenNotPaused {
        _deposit(msg.sender, account, amount);
```

```
61      }
62
63      /// @dev References the WithdrawalInfo for how much the user is permitted to withdraw
64      /// @dev No withdrawal permitted unless currentCycle >= minCycle
65      /// @dev Decrements withheldLiquidity by the withdrawn amount
66      /// @dev TODO Update rewardsContract with proper accounting
67      function withdraw(uint256 requestedAmount) external override whenNotPaused nonReentran
68          require(
69              requestedAmount <= requestedWithdrawals[msg.sender].amount,
70              "WITHDRAW_INSUFFICIENT_BALANCE"
71          );
72          require(requestedAmount > 0, "NO_WITHDRAWAL");
73          require(underlyer.balanceOf(address(this)) >= requestedAmount, "INSUFFICIENT_POOL_
74
75          // Checks for manager cycle and if user is allowed to withdraw based on their mini
76          require(
77              requestedWithdrawals[msg.sender].minCycle <= manager.getCurrentCycleIndex(),
78              "INVALID_CYCLE"
79          );
80
81          requestedWithdrawals[msg.sender].amount = requestedWithdrawals[msg.sender].amount.
82              requestedAmount
83          );
84
85          // If full amount withdrawn delete from mapping
86          if (requestedWithdrawals[msg.sender].amount == 0) {
87              delete requestedWithdrawals[msg.sender];
88          }
89
90          withheldLiquidity = withheldLiquidity.sub(requestedAmount);
91
92          _burn(msg.sender, requestedAmount);
93          underlyer.safeTransfer(msg.sender, requestedAmount);
94
95          bytes32 eventSig = "Withdraw";
96          encodeAndSendData(eventSig, msg.sender);
97      }
98
99      /// @dev Adjusts the withheldLiquidity as necessary
100     /// @dev Updates the WithdrawalInfo for when a user can withdraw and for what requested
101     function requestWithdrawal(uint256 amount) external override {
102         require(amount > 0, "INVALID_AMOUNT");
103         require(amount <= balanceOf(msg.sender), "INSUFFICIENT_BALANCE");
104
105         //adjust withheld liquidity by removing the original withheld amount and adding the
106         withheldLiquidity = withheldLiquidity.sub(requestedWithdrawals[msg.sender].amount)
107             amount
108         );
109         requestedWithdrawals[msg.sender].amount = amount;
110         if (manager.getRolloverStatus()) { // If manger is currently rolling over add two
111             requestedWithdrawals[msg.sender].minCycle = manager.getCurrentCycleIndex().add
112         } else { // If manager is not rolling over add one to minimum withdrawal cycle
113             requestedWithdrawals[msg.sender].minCycle = manager.getCurrentCycleIndex().add
```

```
        }

        emit WithdrawalRequested(msg.sender, amount);
    }

    function preTransferAdjustWithheldLiquidity(address sender, uint256 amount) internal {
        if (requestedWithdrawals[sender].amount > 0) {
            //reduce requested withdraw amount by transferred amount;
            uint256 newRequestedWithdrawl = requestedWithdrawals[sender].amount.sub(
                Math.min(amount, requestedWithdrawals[sender].amount)
            );

            //subtract from global withheld liquidity (reduce) by removing the delta of (r
            withheldLiquidity = withheldLiquidity.sub(
                requestedWithdrawals[sender].amount.sub(newRequestedWithdrawl)
            );

            //update the requested withdraw for user
            requestedWithdrawals[sender].amount = newRequestedWithdrawl;

            //if the withdraw request is 0, empty it out
            if (requestedWithdrawals[sender].amount == 0) {
                delete requestedWithdrawals[sender];
            }
        }
    }
```

**SLOC Appendix**

Solidity Contracts

| Language | Files | Lines | Blanks | Comments | Code | Complex |
|---|---|---|---|---|---|---|
| Solidity | 9 | 2354 | 435 | 132 | 1787 | 228 |

Comments to Code 132/1787 = 7%

Javascript Tests

| Language | Files | Lines | Blanks | Comments | Code | Complex |
|---|---|---|---|---|---|---|
| JavaScript | N/A | N/A | N/A | N/A | N/A | N/A |

Tests to Code = N/A