0.7

Popsicle Finance Process Quality Review

Score: 41%

Overview

This is a Popsicle Finance Process Quality Review completed on August 10th 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 41%, 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

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such

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 Smart Chain

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra

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://docs.popsicle.finance/our-tech/our-contracts, 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 *SorbettoFragola.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/Popsicle-Finance/Contracts.

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 26 commits and 1 branch, this is an unhealthy software repository.

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:** No

Team uses aliases/personas.

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://docs.popsicle.finance/.

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

> ⚠ **Answer:** No

There are no software functions documented in the Popsicle Finance 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 Popsicle Finance 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:** 85%

Code examples are in the Appendix. As per the SLOC, there is 85% commenting to code (CtC).

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

Note: Interface, library, and any third party files were not used in our calculations. Only the core files.

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%

There are no software functions documented in the Popsicle Finance documentation. Therefore, we cannot evaluate their traceability to the 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 testing suite in the Popsicle Finance GitHub.

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:** 0%

There is no evidence of Popsicle Finance code coverage in any of their documentation or in their audit reports by PeckShield and Certik.

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

As there is no testing suite in the Popsicle Finance GitHub, there is also a lack of scripts or instructions to run tests.

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%

No test result report was found in any of the Popsicle Finance 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%

There is no evidence of a Popsicle Finance Formal Verification test in any of their documentation or in further web research.

16) Stress Testing environment (%)

> ⚠ **Answer:** 0%

There is no evidence of any Popsicle Finance 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:** 70%

Certik has published a Popsicle Finance audit report for their core contracts on May 5th 2021.

PeckShield has published a Popsicle Finance audit report for their SorbettoFragola ecosystem on June 28th 2021.

Both audits were released after the launch of the specific contracts.

Note: Most fix recommendations were implemented by the Popsicle Finance team.

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:** 70%

Popsicle Finance has a Bug Bounty Program with immuneFi of 100,000$.

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:** 100%

There is a clearly labelled "Governance" section in the Popsicle Finance documentation.

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:** 30%

b) Defined roles are described in the governance section of their documentation, as well as a multisig function in this article.

Note: Contracts are not clearly labelled as upgradeable or not. Also, their governance portal link is dead.

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%

Although all governance-related information is detailed in user-friendly language, the information does not relate to investments safety.

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%

No evidence of a Pause Control or similar function was found in any of the Popsicle Finance documentation.

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	Popsicle Finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	260		106.25
Code and Team			**41%**
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	n	0
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%) c	15	0%	0
9) Are there sufficiently detailed comments for all function	5	85%	4.25
10) Is it possible to trace from software documentation to th	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	5	0%	0
13) Scripts and instructions to run the tests? (Y/N)	5	0	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	70%	49
18) Is the bug bounty acceptable high? (%)	10	70%	7
Access Controls			
19) Can a user clearly and quickly find the status of the admi	5	100%	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	10	0%	0
Section Scoring			

Code and Team	50	60%	
Documentation	45	21%	
Testing	50	0%	
Security	80	70%	
Access Controls	35	31%	

Executing Code Appendix

Popsicle Finance Contracts

You can find all of our contracts on our GitHub Page **here**.

fUSDt Contracts

fUSDt **token contract** on Binance Smart Chain.

fUSDt **token contract** on Fantom Blockchain.

Fragola Contracts

USDC/ETH (0.3%)

Pool: 0x8ad599c3A0ff1De082011EFDDc58f1908eb6e6D8
Fragola: 0xd63b340F6e9CCcF0c997c83C8d036fa53B113546

WBTC/ETH (0.3%)

Pool: 0xCBCdF9626bC03E24f779434178A73a0B4bad62eD
Fragola: 0x0A8143EF65b0CE4C2fAD195165ef13772ff6Cca0

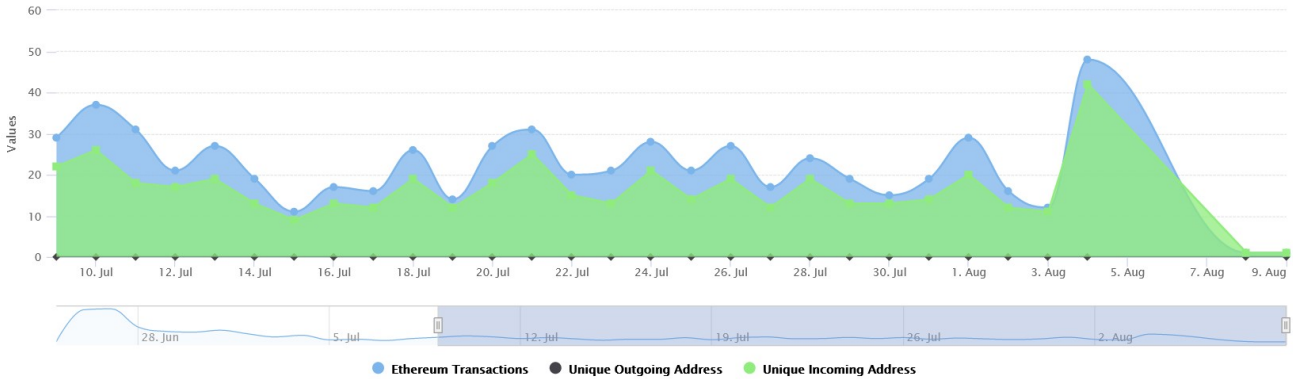
ETH/USDT(0.3%)

Code Used Appendix

Ether Transactions for 0xd63b340F6e9CCcF0c997c83C8d036fa53B113546
Source: Etherscan.io

Zoom 1m 6m 1y All

From Jul 9, 2021 To Aug 9, 2021



Example Code Appendix

```
/// @title Sorbetto Fragola is a yield enchancement v3 contract
/// @dev Sorbetto fragola is a Uniswap V3 yield enchancement contract which acts as
/// intermediary between the user who wants to provide liquidity to specific pools
/// and earn fees from such actions. The contract ensures that user position is in
/// range and earns maximum amount of fees available at current liquidity utilization
/// rate.
contract SorbettoFragola is ERC20Permit, ReentrancyGuard, ISorbettoFragola {
    using LowGasSafeMath for uint256;
    using LowGasSafeMath for uint160;
    using LowGasSafeMath for uint128;
    using UnsafeMath for uint256;
    using SafeCast for uint256;
    using PoolVariables for IUniswapV3Pool;
    using PoolActions for IUniswapV3Pool;

    //Any data passed through by the caller via the IUniswapV3PoolActions#mint call
    struct MintCallbackData {
        address payer;
    }
    //Any data passed through by the caller via the IUniswapV3PoolActions#swap call
    struct SwapCallbackData {
        bool zeroForOne;
    }
    // Info of each user
    struct UserInfo {
        uint256 token0Rewards; // The amount of fees in token 0
        uint256 token1Rewards; // The amount of fees in token 1
        uint256 token0PerSharePaid; // Token 0 reward debt
        uint256 token1PerSharePaid; // Token 1 reward debt
    }

    /// @notice Emitted when user adds liquidity
    /// @param sender The address that minted the liquidity
    /// @param liquidity The amount of liquidity added by the user to position
    /// @param amount0 How much token0 was required for the added liquidity
    /// @param amount1 How much token1 was required for the added liquidity
    event Deposit(
        address indexed sender,
```

```
        uint256 liquidity,
        uint256 amount0,
        uint256 amount1
    );

    /// @notice Emitted when user withdraws liquidity
    /// @param sender The address that minted the liquidity
    /// @param shares of liquidity withdrawn by the user from the position
    /// @param amount0 How much token0 was required for the added liquidity
    /// @param amount1 How much token1 was required for the added liquidity
    event Withdraw(
        address indexed sender,
        uint256 shares,
        uint256 amount0,
        uint256 amount1
    );

    /// @notice Emitted when fees was collected from the pool
    /// @param feesFromPool0 Total amount of fees collected in terms of token 0
    /// @param feesFromPool1 Total amount of fees collected in terms of token 1
    /// @param usersFees0 Total amount of fees collected by users in terms of token 0
    /// @param usersFees1 Total amount of fees collected by users in terms of token 1
    event CollectFees(
        uint256 feesFromPool0,
        uint256 feesFromPool1,
        uint256 usersFees0,
        uint256 usersFees1
    );

    /// @notice Emitted when sorbetto fragola changes the position in the pool
    /// @param tickLower Lower price tick of the positon
    /// @param tickUpper Upper price tick of the position
    /// @param amount0 Amount of token 0 deposited to the position
    /// @param amount1 Amount of token 1 deposited to the position
    event Rerange(
        int24 tickLower,
        int24 tickUpper,
        uint256 amount0,
        uint256 amount1
    );

    /// @notice Emitted when user collects his fee share
    /// @param sender User address
    /// @param fees0 Exact amount of fees claimed by the users in terms of token 0
    /// @param fees1 Exact amount of fees claimed by the users in terms of token 1
    event RewardPaid(
        address indexed sender,
        uint256 fees0,
        uint256 fees1
    );

    /// @notice Shows current Sorbetto's balances
    /// @param totalAmount0 Current token0 Sorbetto's balance
```

```
92
    /// @param totalAmount1 Current token1 Sorbetto's balance
93  event Snapshot(uint256 totalAmount0, uint256 totalAmount1);
94
95  event TransferGovernance(address indexed previousGovernance, address indexed newGoverna
96
97  /// @notice Prevents calls from users
98  modifier onlyGovernance {
99      require(msg.sender == governance, "OG");
100     _;
101 }
102
103 mapping(address => UserInfo) public userInfo; // Info of each user that provides liqui
104 /// @inheritdoc ISorbettoFragola
105 address public immutable override token0;
106 /// @inheritdoc ISorbettoFragola
107 address public immutable override token1;
108 // WETH address
109 address public immutable weth = 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;
110 // @inheritdoc ISorbettoFragola
111 int24 public immutable override tickSpacing;
112 uint24 immutable GLOBAL_DIVISIONER = 1e6; // for basis point (0.0001%)
113
114 // @inheritdoc ISorbettoFragola
115 IUniswapV3Pool public override pool;
116 // Accrued protocol fees in terms of token0
117 uint256 public accruedProtocolFees0;
118 // Accrued protocol fees in terms of token1
119 uint256 public accruedProtocolFees1;
120 // Total lifetime accrued users fees in terms of token0
121 uint256 public usersFees0;
122 // Total lifetime accrued users fees in terms of token1
123 uint256 public usersFees1;
124 // intermediate variable for user fee token0 calculation
125 uint256 public token0PerShareStored;
126 // intermediate variable for user fee token1 calculation
127 uint256 public token1PerShareStored;
128
129 // Address of the Sorbetto's owner
130 address public governance;
131 // Pending to claim ownership address
132 address public pendingGovernance;
133 //Sorbetto fragola settings address
134 address public strategy;
135 // Current tick lower of sorbetto pool position
136 int24 public override tickLower;
137 // Current tick higher of sorbetto pool position
138 int24 public override tickUpper;
139 // Checks if sorbetto is initialized
140 bool public finalized;
141
142 /**
143  * @dev After deploying, strategy can be set via `setStrategy()`
144  * @param  pool Underlying Uniswap V3 pool with fee = 3000
```

```
145
         * @param _strategy Underlying Sorbetto Strategy for Sorbetto settings
146      */
147      constructor(
148         address _pool,
149         address _strategy
150     ) ERC20("Popsicle LP V3 USDT/WETH", "PLP") ERC20Permit("Popsicle LP V3 USDT/WETH") {
151         pool = IUniswapV3Pool(_pool);
152         strategy = _strategy;
153         token0 = pool.token0();
154         token1 = pool.token1();
155         tickSpacing = pool.tickSpacing();
156         governance = msg.sender;
157     }
158     //initialize strategy
159     function init() external onlyGovernance {
160         require(!finalized, "F");
161         finalized = true;
162         int24 baseThreshold = tickSpacing * ISorbettoStrategy(strategy).tickRangeMultiplie
163         ( , int24 currentTick, , , , , ) = pool.slot0();
164         int24 tickFloor = PoolVariables.floor(currentTick, tickSpacing);
165
166         tickLower = tickFloor - baseThreshold;
167         tickUpper = tickFloor + baseThreshold;
168         PoolVariables.checkRange(tickLower, tickUpper); //check ticks also for overflow/un
169     }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	9	7424	893	3000	3531	397

Comments to Code 3000/3531 = 85%

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