Uniswap V3 PQ Review

Score: 94%

Overview

This is a Uniswap Process Quality Review completed on 18 May, 2021. It was performed using the Process Review process (version 0.7) and is documented here. Uniswap V2 is reviewed here. The review was performed by Rex of DeFiSafety. Check out our Telegram.

The final score of the review is 94%, a awesome score. 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**

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: 100%

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

They are available at website https://github.com/Uniswap/uniswap-v3-periphery/blob/main/deploys.md as indicated in the Appendix.

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 35,755 transactions a day on contract *swaprouter.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/Uniswap/uniswap-v3-core

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: 100%

With 933 commits and 1 branch, this is clearly a well-maintained 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

The names of the contract engineers can be found on their GitHub.

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://uniswap.org/whitepaper-v3.pdf

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

Answer: Yes

location: https://docs.uniswap.org/reference/smart-contracts

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: 100%

All the contracts and the functions are clearly well-documented.

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: 49%

Code examples are in the Appendix. As per the SLOC, there is 49% 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: 100%

there is clear explicit traceability between the code and documentation at all levels of the 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 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 Test to Code Ratio of 583%, there is clearly a robust test suite.

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: 50%

There doesn't seem to be any indication of code coverage but obviously with the tests, coverage is considerable.

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/Uniswap/uniswap-v3-core

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: 90%

The results of the unit tests, lineter and mythX are available through the github. No coverage report, but they may not have considered coverage necessary.

https://github.com/Uniswap/uniswap-v3-core/runs/2502426910?check_suite_focus=true

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: 100%

As all testing is based on the spec requirements, this appears to meet the essence of formal verification.

16) Stress Testing environment (%)

> Answer: 100%

This protocol has been stress-tested on all testnets.

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%

ABDK consulting did an audit on the 23rd of march, 2021.

Trail of Bits did an audit on the 12th of March, 2021.

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: 80%

Bug Bounty Location: https://github.com/Uniswap/uniswap-v3-core/blob/main/bug-bounty.md

The maximum bug bounty is 500,000$

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
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%

Location: https://uniswap.org/docs/v2/governance/overview/ and https://uniswap.org/whitepaper-v3.pdf Section 4

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%

All contracts are clearly labelled as upgradeable (or not) -- 10% -- Because the actual control over the protocol by UNI holders is not clearly specified. Are the contracts immutable?

The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% -- UNI holders via a contract

The capabilities for change in the contracts are described a bit but overall upgradeability is not clear.-- 10%

Score: 10+30+10 or 50%

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 -- 25%

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%

The information is given in clear, non-technical terms that pertain to the investments in the Uniswap whitepaper.

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: 100%

The whitepaper (Section 4) indicates the owner cannot pause the contracts, so 100%

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	Uniswap V3	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		244.45
Code and Team			**94%**
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**	100%	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**	100%	15
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	49%	2.45
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	100%	10
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**	50%	2.5
13) Scripts and instructions to run the tests? (Y/N)	**5**	Y	5
14) Report of the results (%)	**10**	90%	9
15) Formal Verification test done (%)	**5**	100%	5
16) Stress Testing environment (%)	**5**	100%	5

Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	80%	8
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**	50%	5
21) Is the information in non-technical terms	**10**	90%	9
22) Is there Pause Control documentation including records of tests	**10**	100%	10
Section Scoring			
Code and Team	50	100%	
Documentation	45	94%	
Testing	50	93%	
Security	80	98%	
Access Controls	35	79%	

Executing Code Appendix

The latest version of `@uniswap/v3-core`, `@uniswap/v3-periphery` are deployed to Ethereum mainnet and all testnets at the same addresses.

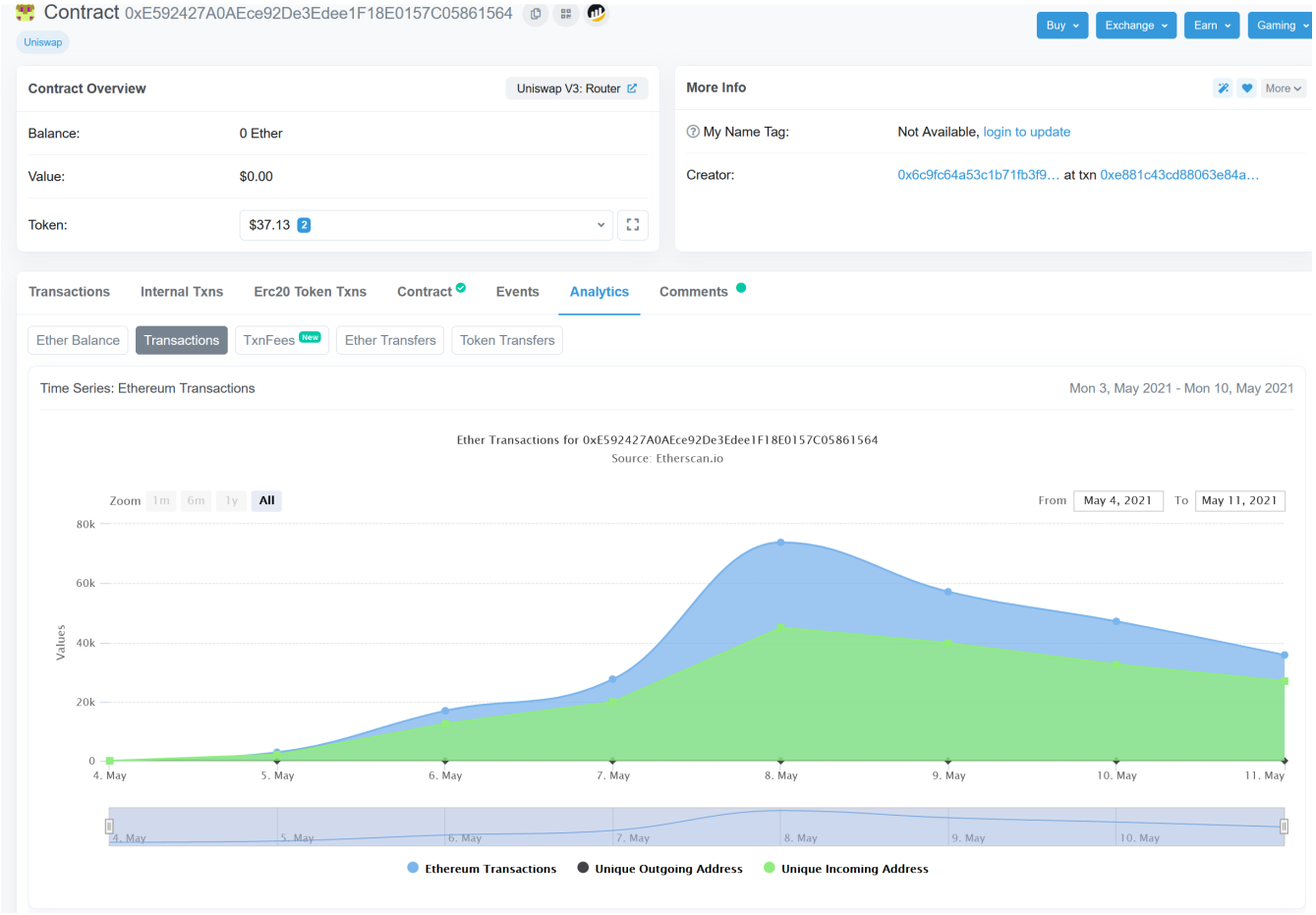
The source code is verified with Etherscan on all networks, for all contracts except `UniswapV3Pool`. We are working on getting the `UniswapV3Pool` contract verified with Etherscan.

These addresses are final and were deployed from these npm package versions:

- `@uniswap/v3-core`: `1.0.0`
- `@uniswap/v3-periphery`: `1.0.0`

Contract	Address	Source Code
UniswapV3Factory	`0x1F98431c8aD98523631AE4a59f267346ea31F984`	https://github.com/Uniswap/uniswap-v3-core/blob/contracts/UniswapV3Factory.sol
Multicall2	`0x5BA1e12693Dc8F9c48aAD8770482f4739bEeD696`	https://etherscan.io/address/0x5BA1e12693Dc8F9c48aAD8770482f4739bEeD69
ProxyAdmin	`0xB753548F6E010e7e680BA186F9Ca1BdAB2E90cf2`	https://github.com/OpenZeppelin/openzeppelin-cc/blob/v3.4.1-solc-0.7-2/contracts/proxy/ProxyAdmi
TickLens	`0xbfd8137f7d1516D3ea5cA83523914859ec47F573`	https://github.com/Uniswap/uniswap-v3-periphery/blob/v1.0.0/contracts/lens/TickLens.sol
Quoter	`0xb27308f9F90D607463bb33eA1BeBb41C27CE5AB6`	https://github.com/Uniswap/uniswap-v3-periphery/blob/v1.0.0/contracts/lens/Quoter.sol
SwapRouter	`0xE592427A0AEce92De3Edee1F18E0157C05861564`	https://github.com/Uniswap/uniswap-v3-periphery/blob/v1.0.0/contracts/SwapRouter.sol
NFTDescriptor	`0x42B24A95702b9986e82d421cC3568932790A48Ec`	https://github.com/Uniswap/uniswap-v3-periphery/blob/v1.0.0/contracts/libraries/NFTDescriptor.sol
NonfungibleTokenPositionDescriptor	`0x91ae842A5Ffd8d12023116943e72A606179294f3`	https://github.com/Uniswap/uniswap-v3-periphery/blob/v1.0.0/contracts/NonfungibleTokenPositionDescriptor.sol
TransparentUpgradeableProxy	`0xEe6A57eC80ea46401049E92587E52f5Ec1c24785`	https://github.com/OpenZeppelin/openzeppelin-cc/blob/v3.4.1-solc-0.7-2/contracts/proxy/TransparentUpgradeableProxy.sol
NonfungiblePositionManager	`0xC36442b4a4522E871399CD717aBDD847Ab11FE88`	https://github.com/Uniswap/uniswap-v3-periphery/blob/v1.0.0/contracts/NonfungiblePositionManage
V3Migrator	`0xA5644E29708357803b5A882D272c41cC0dF92B34`	https://github.com/Uniswap/uniswap-v3-periphery/blob/v1.0.0/contracts/V3Migrator.sol

Code Used Appendix



Example Code Appendix

```
1  // SPDX-License-Identifier: BUSL-1.1
2  pragma solidity =0.7.6;
3
4  import './interfaces/IUniswapV3Factory.sol';
5
6  import './UniswapV3PoolDeployer.sol';
7  import './NoDelegateCall.sol';
8
9  import './UniswapV3Pool.sol';
10
11 /// @title Canonical Uniswap V3 factory
12 /// @notice Deploys Uniswap V3 pools and manages ownership and control ov
13 contract UniswapV3Factory is IUniswapV3Factory, UniswapV3PoolDeployer, No
14     /// @inheritdoc IUniswapV3Factory
15     address public override owner;
16
17     /// @inheritdoc IUniswapV3Factory
18     mapping(uint24 => int24) public override feeAmountTickSpacing;
19     /// @inheritdoc IUniswapV3Factory
20     mapping(address => mapping(address => mapping(uint24 => address))) pu
```

```
21
22      constructor() {
23          owner = msg.sender;
24          emit OwnerChanged(address(0), msg.sender);
25
26          feeAmountTickSpacing[500] = 10;
27          emit FeeAmountEnabled(500, 10);
28          feeAmountTickSpacing[3000] = 60;
29          emit FeeAmountEnabled(3000, 60);
30          feeAmountTickSpacing[10000] = 200;
31          emit FeeAmountEnabled(10000, 200);
32      }
33
34      /// @inheritdoc IUniswapV3Factory
35      function createPool(
36          address tokenA,
37          address tokenB,
38          uint24 fee
39      ) external override noDelegateCall returns (address pool) {
40          require(tokenA != tokenB);
41          (address token0, address token1) = tokenA < tokenB ? (tokenA, tok
42          require(token0 != address(0));
43          int24 tickSpacing = feeAmountTickSpacing[fee];
44          require(tickSpacing != 0);
45          require(getPool[token0][token1][fee] == address(0));
46          pool = deploy(address(this), token0, token1, fee, tickSpacing);
47          getPool[token0][token1][fee] = pool;
48          // populate mapping in the reverse direction, deliberate choice t
49          getPool[token1][token0][fee] = pool;
50          emit PoolCreated(token0, token1, fee, tickSpacing, pool);
51      }
52
53      /// @inheritdoc IUniswapV3Factory
54      function setOwner(address _owner) external override {
55          require(msg.sender == owner);
56          emit OwnerChanged(owner, _owner);
57          owner = _owner;
58      }
59
60      /// @inheritdoc IUniswapV3Factory
61      function enableFeeAmount(uint24 fee, int24 tickSpacing) public overri
62          require(msg.sender == owner);
63          require(fee < 1000000);
64          // tick spacing is capped at 16384 to prevent the situation where
65          // TickBitmap#nextInitializedTickWithinOneWord overflows int24 con
66          // 16384 ticks represents a >5x price change with ticks of 1 bips
67          require(tickSpacing > 0 && tickSpacing < 16384);
68          require(feeAmountTickSpacing[fee] == 0);
69
70          feeAmountTickSpacing[fee] = tickSpacing;
71          emit FeeAmountEnabled(fee, tickSpacing);
72      }
```

```
73  }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	17	1682	184	495	1003	105

Comments to Code 495/1003 = 49%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	22	6632	688	88	5856	394

Tests to Code 5856/1003 = 583%