

# Curve Finance Process Quality Review

Score 77%

---

This is a [Curve Finance](#) Process Quality Audit completed on 27 January, 2021. It was performed using the Process Review process (version 0.6.1) and is documented [here](#). It is a revision of a [report](#) completed on August 2020. That report was performed using the Process Audit process (version 0.4) and is documented [here](#). The audit was performed by ShinkaRex of [Caliburn Consulting](#). Check out our [Telegram](#).

The final score of the audit is 77%, a clear pass. The breakdown of the scoring is in [Scoring Appendix](#).

## Summary of the Process

Very simply, the audit 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**

## 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, 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. 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.

This completed report is copyright (c) DeFiSafety 2021. Permission is given to copy in whole, retaining this copyright label.

## 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? (Y/N)
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)

### Are the executing code addresses readily available? (Y/N)

 Answer: Yes

The address is at the "?" and then "Contracts". See [Appendix](#).

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

 Answer: 100%

Activity is in excess of 100 token transfers a day, 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


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

 Answer: Yes

GitHub: <https://github.com/curvefi/curve-contract>

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.

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

 Answer: 100%

With 736 commits and 21 branches, this is a healthy repo.

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

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

 Answer: Yes

Charlie is no longer anon.

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;

1. Is there a whitepaper? (Y/N)
2. Are the basic application requirements documented? (Y/N)
3. Do the requirements fully (100%) cover the deployed contracts? (%)
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)
5. Is it possible to trace software requirements to the implementation in code (%)

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



Answer: Yes

Location: <https://www.curve.fi/rootfaq>

## Are the basic application requirements documented? (Y/N)



Answer: No

Location: None

Apart from the whitepaper, no explanation of the contracts are visible.

### How to improve this score

Write the document based on the deployed code. For guidance, refer to the [SecurEth System Description Document](#).

## Do the requirements fully (100%) cover the deployed contracts? (%)



Answer: 0%

With no documentation, there can be no coverage of the code. There is a [requirements.txt](#) file, but it only lists tools used to build.

### 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.

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



Answer: 10%

There is very little commenting in the code. Explanatory comments only appear occasionally.

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 10% commenting to code.

### 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](#).

## Is it possible to trace requirements to the implementation in code (%)



Answer: 0%

With no requirements, it is impossible to trace them to 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;

1. Full test suite (Covers all the deployed code) (%)
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)
3. Scripts and instructions to run the tests (Y/N)
4. Packaged with the deployed code (Y/N)
5. Report of the results (%)
6. Formal Verification test done (%)
7. Stress Testing environment (%)

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

✓ Answer: 100%

There is a test suite. There is a 170% test to code ratio, but that only considers the compound swap code and the full package is larger. The test appear to cover functional rather than unit tests.

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.

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

! Answer: 50%

Clearly some amount of coverage was gained from the test that were run, but no test results are available. Score is midfield on assumption of code coverage.

### 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.

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

✓ Answer: Yes

While there are scripts for environment setup, no scripts for the test environment are evident.

### 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.

### Packaged with the deployed code (Y/N)

✓ Answer: Yes

Testing integrated with virtual environment setup script (<https://github.com/curvefi/curve-contract/blob/master/install.sh>)

### Report of the results (%)

i Answer: 70%



A test report was part of the [Quantstamp](#) audit. The report indicated all test results but gave no indication of code coverage.

### How to improve this score

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

## Formal Verification test done (%)



Answer: 0%

There is no evidence of Formal Validation tests on this codebase. In their defense, this is still a rare type of test to include.

## Stress Testing environment (%)



Answer: 0%

No test nets evident.

## Audits



Answer: 100%

Curve Finance had a number of security audits evident from [Quantstamp](#) and [Trail of Bits](#). Good audit quality is evident in the Quantstamp audit especially.

1. Multiple Audits performed before deployment and results public and

- implemented or not required (100%)
- 2. Single audit performed before deployment and results public and implemented or not required (90%)
- 3. Audit(s) performed after deployment and no changes required. Audit report is public. (70%)
- 4. No audit performed (20%)
- 5. Audit Performed after deployment, existence is public, report is not public and no improvements deployed (0%)

## Appendices

### Author Details

The author of this audit is Rex of [Caliburn Consulting](#).

Email : [rex@caliburnc.com](mailto:rex@caliburnc.com) Twitter : @ShinkaRex

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 2017 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 Audits are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

Career wise I am a business development for an avionics supplier.


### Scoring Appendix

PQ Audit Scoring Matrix (v0.6)	Total	Curve rev	
	Points	Answer	Points
Total	240		185.5
<b>Code and Team</b>			<b>77%</b>
2. Are the basic software functions documented? (Y/N)	10	N	0


3. Does the software function documentation fully (100%) cover the deployed contracts? (%)	15	0%	0
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)	10	10%	1
5 Is it possible to trace from software documentation to the implementation in code (%)	5	0%	0
<b>Testing</b>			
1. Full test suite (Covers all the deployed code) (%)	20	100%	20
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	50%	2.5
3. Scripts and instructions to run the tests? (Y/N)	5	Y	5
4. Packaged with the deployed code (Y/N)	5	Y	5
5. Report of the results (%)	10	70%	7
6. Formal Verification test done (%)	5	0%	0
7. Stress Testing environment (%)	5	0%	0
<b>Audits</b>			
Audit done	70	100%	70
<b>Section Scoring</b>			
Code and Team	70	100%	
Documentation	45	13%	
Testing	55	72%	
Audits	70	100%	
<b>Audit Number</b>			
		66	
<b>Date</b>			
		25-Jan-21	
<b>Private Repo</b>			
		Y	
<b>Version</b>			
		0.6	

## Executing Code Appendix


[←](#)
[→](#)
[↺](#)
[github.com/curvefi/curve-vue/blob/master/src/docs/README.md#how-to-integrate-curve-smart-contracts](#)



**curvefi / curve-vue**  
 forked from pengiundev/curve-vue

[Code](#)
[Pull requests 1](#)
[Actions](#)
[Projects](#)
[Wiki](#)
[Security](#)
[Insights](#)


 master

curve-vue / src / docs / README.md


 pengiundev Update README with remove\_liquidity\_one\_coin function


 1 contributor

399 lines (280 sloc) | 14.3 KB

### Curve pools

- compound(cDAI, cUSDC)
- usdt(cDAI, cUSDC, USDT)
- y(yDAI, yUSDC, yUSDT, yTUSD)
- busd(yDAI, yUSDC, yUSDT, yBUSD)
- susd(DAI, USDC, USDT, sUSD)
- ren(renBTC, wBTC)
- sbtc(renBTC, wBTC, sBTC)

Compound pool

[Swap address](#)

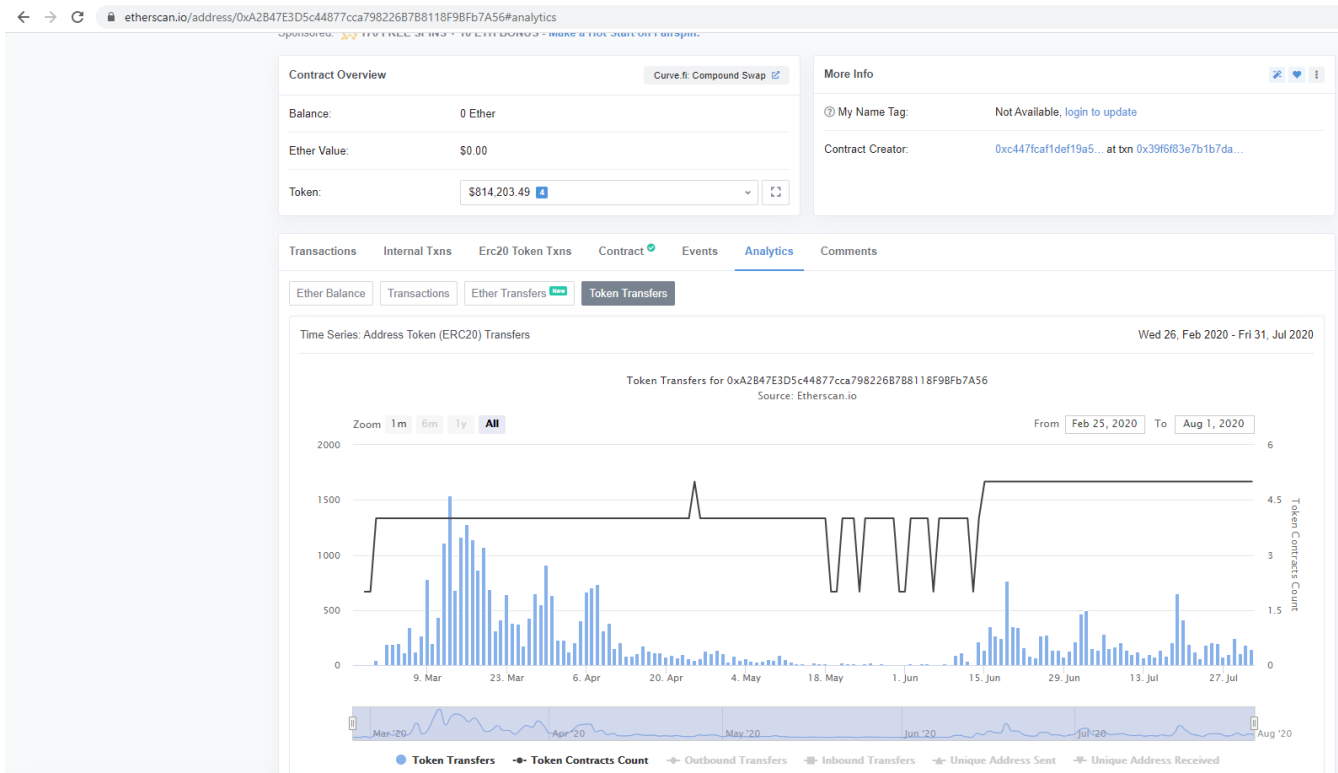
Underlying coins:

1. DAI
2. USDC

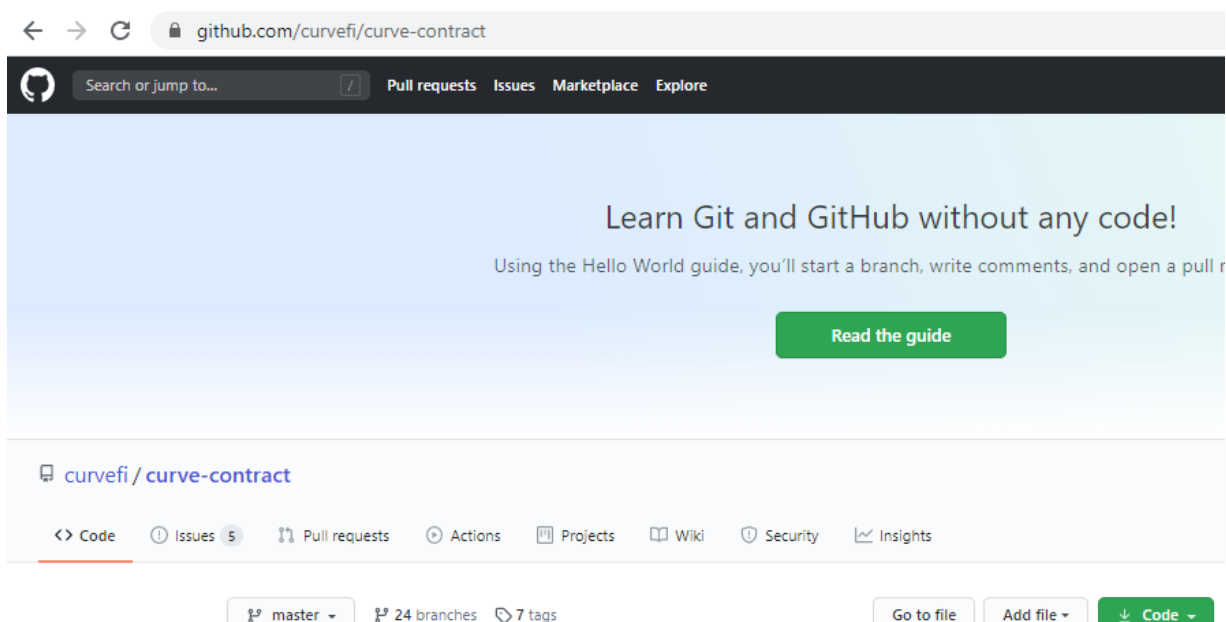
Coins:


1. cDAI
2. cUSDC

## Code Used Appendix



## Repository Healthy Appendix



 michwill	Readme with links to pools	2b8ff42 on Feb 19	🕒 249 commits
📁 deployed	Fixed contract	6 months ago	
📁 tests	Error in get_dx calculation fixed	6 months ago	
📁 vyper	Error in get_dx calculation fixed	6 months ago	
📄 .gitignore	Update ignore list	7 months ago	
📄 README.rst	Readme with links to pools	6 months ago	
📄 deploy-dev.py	Deploy is deploy-dev now	7 months ago	
📄 deploy-mainnet.py.3way	Gas management	6 months ago	
📄 deploy-mainnet.py.template	Change name of the token	6 months ago	
📄 deploy-rinkeby.py	cERC20 interface in deployment script	7 months ago	
📄 etherscanify.py	Update etherscanify	6 months ago	
📄 install.sh	Setting up the environment	11 months ago	
📄 integrations.md	Integrations updated	6 months ago	
📄 requirements.txt	New dependencies, new vyper	7 months ago	

## Example Code Appendix

```

1  # (c) Curve.Fi, 2020
2
3  import ERC20m as ERC20m
4  import cERC20 as cERC20
5  from vyper.interfaces import ERC20
6
7
8  # Tether transfer-only ABI
9  contract USDT:
10     def transfer(_to: address, _value: uint256): modifying
11     def transferFrom(_from: address, _to: address, _value: uint256): modi
12
13
14  # This can (and needs to) be changed at compile time
15  N_COINS: constant(int128) = __N_COINS__ # <- change
16
17  ZERO256: constant(uint256) = 0 # This hack is really bad XXX
18  ZEROS: constant(uint256[N_COINS]) = __N_ZEROS__ # <- change
19
20  USE_LENDING: constant(bool[N_COINS]) = __USE_LENDING__
21  TETHERED: constant(bool[N_COINS]) = __TETHERED__
22
23  FEE_DENOMINATOR: constant(uint256) = 10 ** 10
24  PRECISION: constant(uint256) = 10 ** 18 # The precision to convert to
25  PRECISION_MUL: constant(uint256[N_COINS]) = __PRECISION_MUL__
26  # PRECISION_MUL: constant(uint256[N_COINS]) = [
27  #     PRECISION / convert(PRECISION, uint256), # DAI
28  #     PRECISION / convert(10 ** 6, uint256), # USDC
29  #     PRECISION / convert(10 ** 6, uint256)] # USDT
30

```

```
31
32 admin_actions_delay: constant(uint256) = 3 * 86400
33
34 # Events
35 TokenExchange: event({buyer: indexed(address), sold_id: int128, tokens_so
36 TokenExchangeUnderlying: event({buyer: indexed(address), sold_id: int128,
37 AddLiquidity: event({provider: indexed(address), token_amounts: uint256[N
38 RemoveLiquidity: event({provider: indexed(address), token_amounts: uint25
39 RemoveLiquidityImbalance: event({provider: indexed(address), token_amount
40 CommitNewAdmin: event({deadline: indexed(timestamp), admin: indexed(addre
41 NewAdmin: event({admin: indexed(address)})
42 CommitNewParameters: event({deadline: indexed(timestamp), A: uint256, fee
43 NewParameters: event({A: uint256, fee: uint256, admin_fee: uint256})
44
45 coins: public(address[N_COINS])
46 underlying_coins: public(address[N_COINS])
47 balances: public(uint256[N_COINS])
48 A: public(uint256) # 2 x amplification coefficient
49 fee: public(uint256) # fee * 1e10
50 admin_fee: public(uint256) # admin_fee * 1e10
51 max_admin_fee: constant(uint256) = 5 * 10 ** 9
52
53 owner: public(address)
54 token: ERC20m
55
56 admin_actions_deadline: public(timestamp)
57 transfer_ownership_deadline: public(timestamp)
58 future_A: public(uint256)
59 future_fee: public(uint256)
60 future_admin_fee: public(uint256)
61 future_owner: public(address)
62
63 kill_deadline: timestamp
64 kill_deadline_dt: constant(uint256) = 2 * 30 * 86400
65 is_killed: bool
66
67
68 @public
69 def __init__(_coins: address[N_COINS], _underlying_coins: address[N_COINS
70             _pool_token: address,
71             _A: uint256, _fee: uint256):
72     """
73     _coins: Addresses of ERC20 contracts of coins (c-tokens) involved
74     _underlying_coins: Addresses of plain coins (ERC20)
75     _pool_token: Address of the token representing LP share
76     _A: Amplification coefficient multiplied by n * (n - 1)
77     _fee: Fee to charge for exchanges
78     """
79     for i in range(N_COINS):
80         assert _coins[i] != ZERO_ADDRESS
81         assert _underlying_coins[i] != ZERO_ADDRESS
82         self.balances[i] = 0
```

```
83     self.coins = _coins
84     self.underlying_coins = _underlying_coins
85     self.A = _A
86     self.fee = _fee
87     self.admin_fee = 0
88     self.owner = msg.sender
89     self.kill_deadline = block.timestamp + kill_deadline_dt
90     self.is_killed = False
91     self.token = ERC20m(_pool_token)
```

## SLOC Appendix

### Vyper Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Vyper	52	33141	3652	3129	26360	2127

Comments to Code  $3129 / 26360 = 12\%$

This second list removes the many repeated contracts where only the token changes (from the pools directorty

Language	Files	Lines	Blanks	Comments	Code	Complexity
Vyper	7	2573	255	367	1951	97

Comments to Code  $367 / 1951 = 18\%$

### Python Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
Python	57	4783	1220	230	3333	674

Tests to Code  $3333 / 1951 = 170\%$