0.7

DeFi Swap (Crypto.com)

Score: 50%

Overview

This is a Crypto.com Process Quality Review completed on September 9th 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Mawuli of DeFiSafety. Check out our Telegram.

The final score of the review is **50%**, 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.

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

Chain

This section indicates the blockchain used by this protocol.

> ✓ **Chain:** Ethereum

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:** 0%

They is no evidence of any smart contract addresses in the Crypto.com documentation.

Note: Although the token address is provided, that is not what we look at for this specific metric.

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:** 0%

As there are no published smart contract addresses, there is no code activity.

Note: Although the token address is provided, that is not what we look at for this specific metric.

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/crypto-com/swap-contracts-core.

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:** 100%

With 200 commits and 27 branches, this is a healthy 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

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

> **Answer:** Yes

https://hk.linkedin.com/company/cryptocom.

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

https://blog.crypto.com/defi-swap-whitepaper/.

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

> **Answer:** No

There are no software functions documented in the Crypto.com 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 Crypto.com 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 4% 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%

There are no software functions documented in the Crypto.com documentation. Therefore, we cannot determine the traceability as to their implementation in the source code.

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

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%

Code examples are in the Appendix. As per the SLOC, there is 233% testing to code (TtC).

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 judgment 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

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

> **Answer:** 50%

There is no evidence of code coverage in any of the Crypto.com documentation or audit reports. However,

there is a reasonably complete set of tests.

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:** Yes

There is clear evidence of an up to date set of instructions to run the tests.

https://github.com/crypto-com/swap-contracts-core

14) Report of the results (%)

> ⚠ **Answer:** 0%

There is no evidence of any test result reports in any of the Crypto.com documentation.

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 Formal Verification testing having been done.

16) Stress Testing environment (%)

> ⚠ **Answer:** 0%

Kovan stress-testing is mentioned in GitHub however, no addresses are published and therefore no verification can be done for the stress testing.

https://github.com/crypto-com/swap-contracts-core

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:** 0%

While two valid audits are listed below, as per our guidance without contract addresses, the

Slowmist has published a first Defi Swap audit report on September 4th 2020, which was before their mainnet launch on September 11th 2020.

Slowmist has published a second Defi Swap audit report on September 10th 2020, which was before their mainnet launch on September 11th 2020.

Note: Most fix recommendations were successfully implemented by the Defi Swap 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:** 20%

Bug Bounty value is active at 10K.

https://hackerone.com/crypto?type=team

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:** 0%

Access control could not be found in any the Crypto.com 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:** 0%

There is little to no available information pertaining the mutability or upgradeability of Crypto.com contracts.

Note: The swap contract's code seems to be upgradeable through the Initialize function, and the linking of multiple external interface smart contracts.

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:** 0%

There are currently no access controls documented in the Crypto.com documentations.

Note: The swap contract's code seems to be upgradeable through the Initialize function, and the linking of multiple external interface smart contracts.

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 no pause control or pause control documentation available in the Crypto.com documentation.

Note: The swap contract's code seems to be upgradeable through the Initialize function, and the linking of multiple external interface smart contracts.

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	eFi Swap (Crypto.cor	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		59.5
Code and Team			**23%**
1) Are the executing code addresses readily available? (%)	**20**	0%	0
2) Is the code actively being used? (%)	**5**	0%	0
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**	n	0
8) Does the software function documentation fully (100%) cov	**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**	100%	20
12) Code coverage (Covers all the deployed lines of code, or ex	**5**	50%	2.5
13) Scripts and instructions to run the tests? (Y/N)	**5**	y	5
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**	0%	0
18) Is the bug bounty acceptable high? (%)	**10**	20%	2
Access Controls			
19) Can a user clearly and quickly find the status of the admin	**5**	0%	0
20) Is the information clear and complete	**10**	0%	0
21) Is the information in non-technical terms	**10**	0%	0
22) Is there Pause Control documentation including records of	**10**	0%	0
Section Scoring			
Code and Team	50	50%	
Documentation	45	11%	
Testing	50	55%	
Security	80	3%	
Access Controls	35	0%	

Executing Code Appendix

N/A

Code Used Appendix

N/A

Example Code Appendix

```
contract CroDefiSwapPair is ICroDefiSwapPair, CroDefiSwapERC20 {
    using SafeMath  for uint;
    using UQ112x112 for uint224;

    uint public constant MINIMUM_LIQUIDITY = 10**3;
    bytes4 private constant SELECTOR = bytes4(keccak256(bytes('transfer(address,uint256)')

    address public factory;
    address public token0;
    address public token1;

    uint112 private reserve0;           // uses single storage slot, accessible via getRese
    uint112 private reserve1;           // uses single storage slot, accessible via getRese
    uint32  private blockTimestampLast; // uses single storage slot, accessible via getRese

    uint public price0CumulativeLast;
    uint public price1CumulativeLast;
    uint public kLast; // reserve0 * reserve1, as of immediately after the most recent liqu

    uint private unlocked = 1;
    modifier lock() {
        require(unlocked == 1, 'CroDefiSwap: LOCKED');
        unlocked = 0;
        _;
        unlocked = 1;
    }

    function getReserves() public view returns (uint112 _reserve0, uint112 _reserve1, uint3
        _reserve0 = reserve0;
        _reserve1 = reserve1;
        _blockTimestampLast = blockTimestampLast;
    }

    function _safeTransfer(address token, address to, uint value) private {
        (bool success, bytes memory data) = token.call(abi.encodeWithSelector(SELECTOR, to
        require(success && (data.length == 0 || abi.decode(data, (bool))), 'CroDefiSwap: TR
    }

    event Mint(address indexed sender, uint amount0, uint amount1);
    event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
    event Swap(
        address indexed sender,
        uint amount0In,

```

```
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);

    constructor() public {
        factory = msg.sender;
    }

    // called once by the factory at time of deployment
    function initialize(address _token0, address _token1) external {
        require(msg.sender == factory, 'CroDefiSwap: FORBIDDEN'); // sufficient check
        token0 = _token0;
        token1 = _token1;
    }

    // update reserves and, on the first call per block, price accumulators
    function _update(uint balance0, uint balance1, uint112 _reserve0, uint112 _reserve1) p
        require(balance0 <= uint112(-1) && balance1 <= uint112(-1), 'CroDefiSwap: OVERFLOW
        uint32 blockTimestamp = uint32(block.timestamp % 2**32);
        uint32 timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired
        if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
            // * never overflows, and + overflow is desired
            price0CumulativeLast += uint(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * t
            price1CumulativeLast += uint(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * t
        }
        reserve0 = uint112(balance0);
        reserve1 = uint112(balance1);
        blockTimestampLast = blockTimestamp;
        emit Sync(reserve0, reserve1);
    }

    // if fee is on, mint liquidity equivalent to 1/6th of the growth in sqrt(k)
    function _mintFee(uint112 _reserve0, uint112 _reserve1) private returns (bool feeOn) {
        address feeTo = ICroDefiSwapFactory(factory).feeTo();
        uint feeToBasisPoint = ICroDefiSwapFactory(factory).feeToBasisPoint();

        feeOn = (feeTo != address(0)) && (feeToBasisPoint > 0);
        uint _kLast = kLast; // gas savings
        if (feeOn) {
            if (_kLast != 0) {
                uint rootK = Math.sqrt(uint(_reserve0).mul(_reserve1));
                uint rootKLast = Math.sqrt(_kLast);
                if (rootK > rootKLast) {
                    uint numerator = totalSupply.mul(rootK.sub(rootKLast));
                    uint denominator = rootK.mul(feeToBasisPoint).add(rootKLast);
                    uint liquidity = numerator / denominator;
                    if (liquidity > 0) _mint(feeTo, liquidity);
                }
            }
        } else if (_kLast != 0) {
```

```
97
            kLast = 0;
98          }
99      }
100
101     // this low-level function should be called from a contract which performs important sa
102     function mint(address to) external lock returns (uint liquidity) {
103         (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
104         uint balance0 = IERC20(token0).balanceOf(address(this));
105         uint balance1 = IERC20(token1).balanceOf(address(this));
106         uint amount0 = balance0.sub(_reserve0);
107         uint amount1 = balance1.sub(_reserve1);
108
109         bool feeOn = _mintFee(_reserve0, _reserve1);
110         uint _totalSupply = totalSupply; // gas savings, must be defined here since totalSu
111         if (_totalSupply == 0) {
112             liquidity = Math.sqrt(amount0.mul(amount1)).sub(MINIMUM_LIQUIDITY);
113            _mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQU
114         } else {
115             liquidity = Math.min(amount0.mul(_totalSupply) / _reserve0, amount1.mul(_totalS
116         }
117         require(liquidity > 0, 'CroDefiSwap: INSUFFICIENT_LIQUIDITY_MINTED');
118         _mint(to, liquidity);
119
120         _update(balance0, balance1, _reserve0, _reserve1);
121         if (feeOn) kLast = uint(reserve0).mul(reserve1); // reserve0 and reserve1 are up-to
122         emit Mint(msg.sender, amount0, amount1);
123     }
124
125     // this low-level function should be called from a contract which performs important sa
126     function burn(address to) external lock returns (uint amount0, uint amount1) {
127         (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
128         address _token0 = token0;                                // gas savings
129         address _token1 = token1;                                // gas savings
130         uint balance0 = IERC20(_token0).balanceOf(address(this));
131         uint balance1 = IERC20(_token1).balanceOf(address(this));
132         uint liquidity = balanceOf[address(this)];
133
134         bool feeOn = _mintFee(_reserve0, _reserve1);
135         uint _totalSupply = totalSupply; // gas savings, must be defined here since totalSu
136         amount0 = liquidity.mul(balance0) / _totalSupply; // using balances ensures pro-rat
137         amount1 = liquidity.mul(balance1) / _totalSupply; // using balances ensures pro-rat
138         require(amount0 > 0 && amount1 > 0, 'CroDefiSwap: INSUFFICIENT_LIQUIDITY_BURNED');
139         _burn(address(this), liquidity);
140         _safeTransfer(_token0, to, amount0);
141         _safeTransfer(_token1, to, amount1);
142         balance0 = IERC20(_token0).balanceOf(address(this));
143         balance1 = IERC20(_token1).balanceOf(address(this));
144
145         _update(balance0, balance1, _reserve0, _reserve1);
146         if (feeOn) kLast = uint(reserve0).mul(reserve1); // reserve0 and reserve1 are up-to
147         emit Burn(msg.sender, amount0, amount1, to);
148     }
```

```
149
150
        // this low-level function should be called from a contract which performs important sa
151     function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data) exter
152         require(amount0Out > 0 || amount1Out > 0, 'CroDefiSwap: INSUFFICIENT_OUTPUT_AMOUNT
153         (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
154         require(amount0Out < _reserve0 && amount1Out < _reserve1, 'CroDefiSwap: INSUFFICIEN
155
156         uint balance0;
157         uint balance1;
158         { // scope for _token{0,1}, avoids stack too deep errors
159         address _token0 = token0;
160         address _token1 = token1;
161         require(to != _token0 && to != _token1, 'CroDefiSwap: INVALID_TO');
162         if (amount0Out > 0) _safeTransfer(_token0, to, amount0Out); // optimistically trans
163         if (amount1Out > 0) _safeTransfer(_token1, to, amount1Out); // optimistically trans
164         if (data.length > 0) ICroDefiSwapCallee(to).croDefiSwapCall(msg.sender, amount0Out
165         balance0 = IERC20(_token0).balanceOf(address(this));
166         balance1 = IERC20(_token1).balanceOf(address(this));
167         }
168         uint amount0In = balance0 > _reserve0 - amount0Out ? balance0 - (_reserve0 - amount
169         uint amount1In = balance1 > _reserve1 - amount1Out ? balance1 - (_reserve1 - amount
170         require(amount0In > 0 || amount1In > 0, 'CroDefiSwap: INSUFFICIENT_INPUT_AMOUNT');
171         { // scope for reserve{0,1}Adjusted, avoids stack too deep errors
172         uint magnifier = 10000;
173         uint totalFeeBasisPoint = ICroDefiSwapFactory(factory).totalFeeBasisPoint();
174         uint balance0Adjusted = balance0.mul(magnifier).sub(amount0In.mul(totalFeeBasisPoin
175         uint balance1Adjusted = balance1.mul(magnifier).sub(amount1In.mul(totalFeeBasisPoin
176         // reference: https://uniswap.org/docs/v2/protocol-overview/glossary/#constant-proc
177         require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1).mu
178         }
179
180         _update(balance0, balance1, _reserve0, _reserve1);
181         emit Swap(msg.sender, amount0In, amount1In, amount0Out, amount1Out, to);
182     }
183
184     // force balances to match reserves
185     function skim(address to) external lock {
186         address _token0 = token0; // gas savings
187         address _token1 = token1; // gas savings
188         _safeTransfer(_token0, to, IERC20(_token0).balanceOf(address(this)).sub(reserve0))
189         _safeTransfer(_token1, to, IERC20(_token1).balanceOf(address(this)).sub(reserve1))
190     }
191
192     // force reserves to match balances
193     function sync() external lock {
194         _update(IERC20(token0).balanceOf(address(this)), IERC20(token1).balanceOf(address(t
195     }
196 }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	4	396	55	12	329	53

Comments to Code 12/329 = 4%

Typescript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
TypeScript	5	939	139	32	768	24

Tests to Code 768/329 = 233%