# Adamant Finance Process Quality Review

Score: 53%

## Overview

This is a Adamant Finance Process Quality Review completed on July 8th 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 53%, a Fail. The breakdown of the scoring is in Scoring Appendix. For our purposes, a pass is 70%.

### Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**
- **Here are the admin controls and strategies**

### Disclaimer

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

**Chain**

This section indicates the blockchain used by this protocol.

> ✓ **Chain: Polygon**

Guidance:
Ethereum
Binance  Smart Chain
Polygon

---

## 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%

They are available at website https://adamantfinance.gitbook.io/adamant-finance/contract-links 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 labelling not clear or easy to find |
| 0% | Executing addresses could not be found |

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

✓ Answer: 100%

Activity is 25,000 transactions a day on contract *ERCFund2.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/eepdev

Note: This is not a company repository, but rather the main dev's.

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: 30%

With 36 commits and 1 branch, this is a semi-acceptable software 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 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

No public team or developer information was found.

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://adamantfinance.gitbook.io/adamant-finance/.

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

> ✓ Answer: Yes

Basic software functions are documented in the "Guides" and "Resources" section of https://adamantfinance.gitbook.io/adamant-finance/.

**8) Does the software function documentation fully (100%) cover the deployed contracts? (%)**

Answer: 50%

Major staking functions are documented loosely, across their entire 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 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: 0%

Code examples are in the Appendix.  As per the SLOC, there is 21% 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: 60%

There is some explicit traceability between documentation and code, but most of it is non-explicit.

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: 0%

No testing suite was found in their GitHub repository.

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: 0%

No test for coverage seen in their GitHub repository, and no code coverage found in their audits.

Note: 2/3 audits inaccessible.

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: No

No scripts or instructions to run tests in their GitHub repository.

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 report was found in their GitHub repository.

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%

No evidence of a Adamant Finance Formal Verification test was found in their documentation or on the web.

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

⚠ Answer: 0%

No evidence of test-net smart contract usage in any of their documentation or GitHub repositories.

---

## 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: 90%

Callisto Security published a Adamant Finance audit report on ? (no date specified).

They have a TechRate audit that they removed in anticipation of their Certik audit.

Certik audit is underway.

Note: Most fix recommendations were implemented.

Note 2: Adamant Finance was launched in May 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 |

| | |
|---|---|
| 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, question |

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: 0%

No evidence of a Adamant Finance Bug Bounty program was found.

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 |

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 access controls (%)**

✓ Answer: 100%

Found easily and quickly at https://adamantfinance.gitbook.io/adamant-finance/contract-owner-privileges.

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: 60%

a) Clearly described as upgradeable

c) Capabilities for change, especially in the vault, are described clearly.

Guidance:
All the contracts are immutable -- 100% OR

a) All contracts are clearly labelled as upgradeable (or not) -- 30% AND
b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
c) The capabilities for change in the contracts are described -- 30%

How to improve this score

Create a document that covers the items described above.  An example is enclosed.

**21) Is the information in non-technical terms that pertain to the investments (%)**

ⓘ Answer: 30%

All access control descriptions are exclusively written in very software-specific language.

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 Pause Control or similar function in Adamant Finance's 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

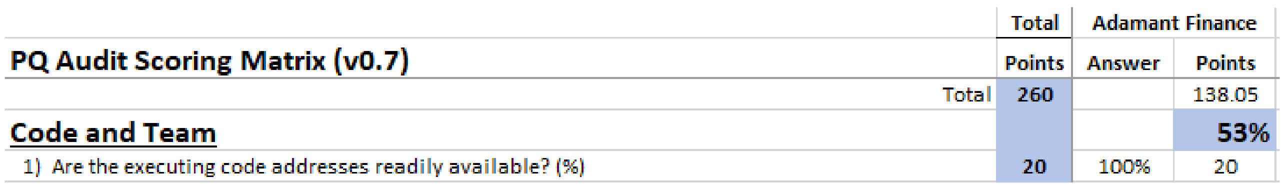
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

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.

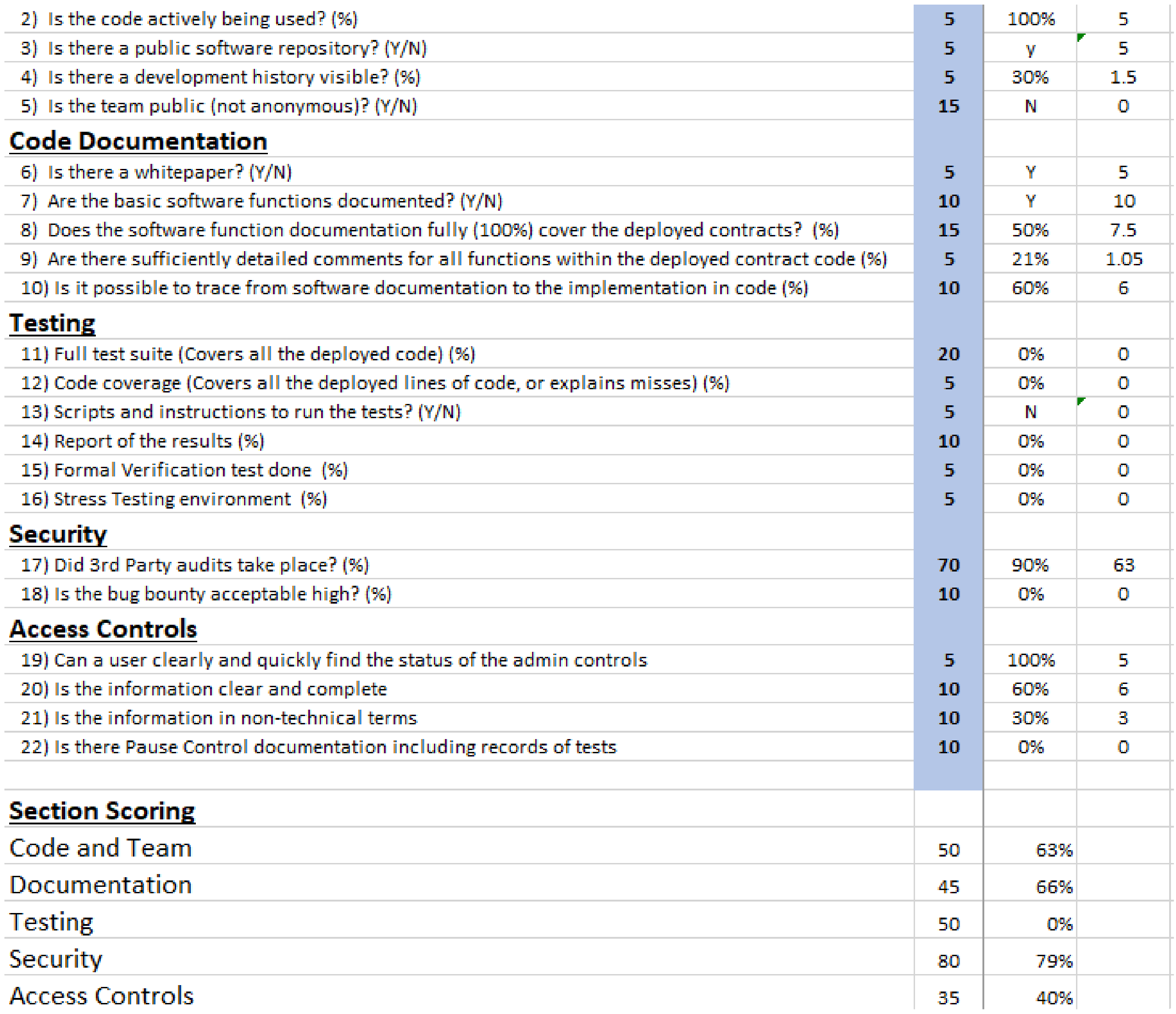
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

### Scoring Appendix



| | Total | Adamant Finance | |
|---|---|---|---|
| **PQ Audit Scoring Matrix (v0.7)** | Points | Answer | Points |
| Total | 260 | | 138.05 |
| **Code and Team** | | | **53%** |
| 1) Are the executing code addresses readily available? (%) | 20 | 100% | 20 |



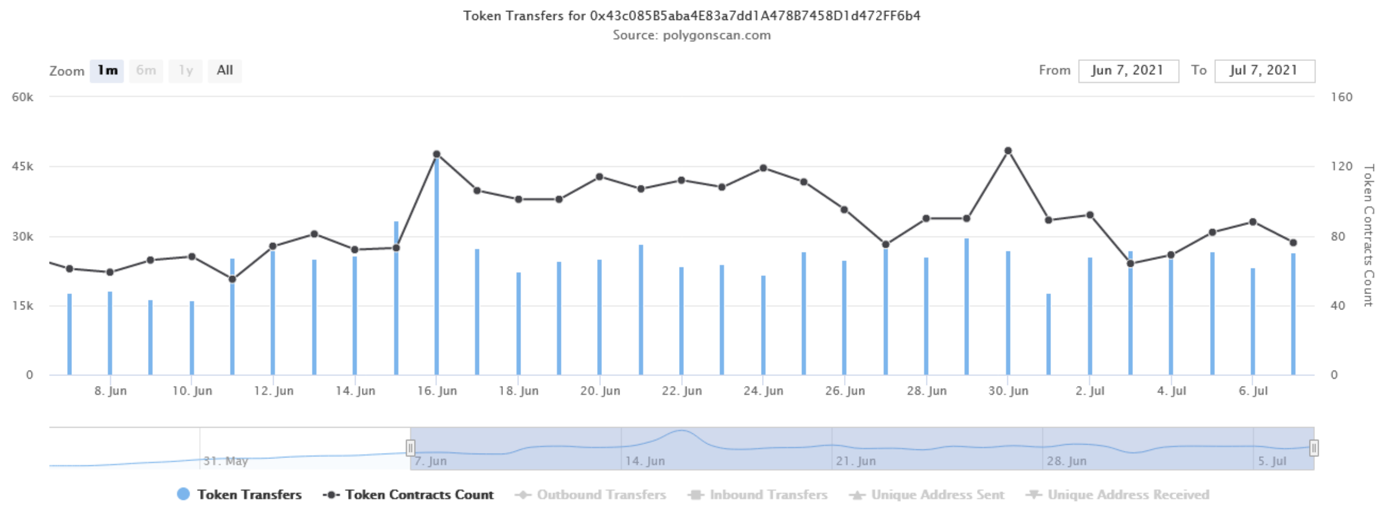
| | | | |
|---|---|---|---|
| 2) Is the code actively being used? (%) | 5 | 100% | 5 |
| 3) Is there a public software repository? (Y/N) | 5 | y | 5 |
| 4) Is there a development history visible? (%) | 5 | 30% | 1.5 |
| 5) Is the team public (not anonymous)? (Y/N) | 15 | N | 0 |
| **Code Documentation** | | | |
| 6) Is there a whitepaper? (Y/N) | 5 | Y | 5 |
| 7) Are the basic software functions documented? (Y/N) | 10 | Y | 10 |
| 8) Does the software function documentation fully (100%) cover the deployed contracts? (%) | 15 | 50% | 7.5 |
| 9) Are there sufficiently detailed comments for all functions within the deployed contract code (%) | 5 | 21% | 1.05 |
| 10) Is it possible to trace from software documentation to the implementation in code (%) | 10 | 60% | 6 |
| **Testing** | | | |
| 11) Full test suite (Covers all the deployed code) (%) | 20 | 0% | 0 |
| 12) Code coverage (Covers all the deployed lines of code, or explains misses) (%) | 5 | 0% | 0 |
| 13) Scripts and instructions to run the tests? (Y/N) | 5 | N | 0 |
| 14) Report of the results (%) | 10 | 0% | 0 |
| 15) Formal Verification test done (%) | 5 | 0% | 0 |
| 16) Stress Testing environment (%) | 5 | 0% | 0 |
| **Security** | | | |
| 17) Did 3rd Party audits take place? (%) | 70 | 90% | 63 |
| 18) Is the bug bounty acceptable high? (%) | 10 | 0% | 0 |
| **Access Controls** | | | |
| 19) Can a user clearly and quickly find the status of the admin controls | 5 | 100% | 5 |
| 20) Is the information clear and complete | 10 | 60% | 6 |
| 21) Is the information in non-technical terms | 10 | 30% | 3 |
| 22) Is there Pause Control documentation including records of tests | 10 | 0% | 0 |
| | | | |
| **Section Scoring** | | | |
| Code and Team | 50 | 63% | |
| Documentation | 45 | 66% | |
| Testing | 50 | 0% | |
| Security | 80 | 79% | |
| Access Controls | 35 | 40% | |

## Executing Code Appendix

### Protocol:

| Name | Address |
|---|---|
| MultiFeeDistribution | 0x920f22E1e5da04504b765F8110ab96A20E6408Bd |
| ERCFund | 0x43c085B5aba4E83a7dd1A478B7458D1d472FF6b4 |
| Minter | 0xAAE758A2dB4204E1334236Acd6E6E73035704921 |
| Calculator | 0xAc2F66971BC37eB443c4E11AAb277e19d1C4864C |
| Timelock | 0x52D3Dcf0E59237B032802b40E69f65877091171F |
| Adamant Token (ADDY) | 0xc3FdbadC7c795EF1D6Ba111e06fF8F16A20Ea539 |

## Code Used Appendix



## Example Code Appendix

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.12;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/SafeERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

import "../interfaces/uniswap/IUniswapV2Pair.sol";
import "../interfaces/uniswap/IUniswapRouterV2.sol";
import "../interfaces/IMultiFeeDistribution.sol";

//Contract where the fees are sent to before they are converted and sent to the feeDistrib
contract ERCFund is Ownable {
    using SafeERC20 for IERC20;
    using SafeMath for uint256;

    address public constant weth = 0x7ceB23fD6bC0adD59E62ac25578270cFf1b9f619; //weth for
    address public currentRouter = 0xa5E0829CaCEd8fFDD4De3c43696c57F7D7A678ff; //Quickswap
    address public feeDistributor;
    bool public feeSharingEnabled = false;
    uint256 public fee = 200;
    uint256 public feeMAX = 10000;

    constructor(address distributor) public {
        feeDistributor = distributor;
    }

    function notifyFeeDistribution(address token) public {
        uint256 balance = IERC20(token).balanceOf(address(this));

        IERC20(token).safeApprove(feeDistributor, 0);
        IERC20(token).safeApprove(feeDistributor, balance);

        IMultiFeeDistribution(feeDistributor).notifyRewardAmount(token, balance);
    }

    //Doesn't support Fee on Transfer tokens, convert those to something else first
    //Transfer token from sender, then transfers it to the fee distributor
    function depositToFeeDistributor(address token, uint256 amount) public {
        IERC20(token).safeTransferFrom(msg.sender, address(this), amount);

        IERC20(token).safeApprove(feeDistributor, 0);
        IERC20(token).safeApprove(feeDistributor, amount);
        IMultiFeeDistribution(feeDistributor).notifyRewardAmount(token, amount);
    }

    /* ========== VIEW FUNCTIONS ========== */

    function feeShareEnabled() external view returns (bool) {
        return feeSharingEnabled;
    }

    function getFee() external view returns (uint256) {
        return fee;
    }

    /* ========== CONVERSION FUNCTIONS ========== */

    function convertFees(address token_in, address token_out) public onlyOwner {
        uint256 balance = IERC20(token_in).balanceOf(address(this));
        if (balance > 0) {
            _swapUniswap(token_in, token_out, balance);
        }
    }

    function convertFeesWithPath(address token_in, address token_out) public onlyOwner {
        uint256 balance = IERC20(token_in).balanceOf(address(this));
        if (balance > 0) {
            address[] memory pair = new address[](2);
            pair[0] = token_in;
            pair[1] = token_out;
            _swapUniswapWithPath(pair, balance);
        }
    }

    function convertFeesWithPathForFeeOnTransferTokens(address token_in, address token_out
        uint256 balance = IERC20(token_in).balanceOf(address(this));
        if (balance > 0) {
            address[] memory pair = new address[](2);
            pair[0] = token_in;
            pair[1] = token_out;
            _swapUniswapWithPathForFeeOnTransferTokens(pair, balance);
        }
    }

    /* ========== SETTER FUNCTIONS ========== */
```

```solidity

    function setFeeDistributor(address distributor) public onlyOwner {
        feeDistributor = distributor;
    }

    function setFeeSharingEnabled(bool enabled) public onlyOwner {
        feeSharingEnabled = enabled;
    }

    function setFee(uint256 _fee) public onlyOwner {
        require(_fee <= 3000);
        fee = _fee;
    }

    /* ========== EMERGENCY FUNCTIONS ========== */

    function recover(address token) public onlyOwner {
        uint256 _token = IERC20(token).balanceOf(address(this));
        if (_token > 0) {
            IERC20(token).safeTransfer(msg.sender, _token);
        }
    }

    /* ========== UNISWAP FUNCTIONS ========== */

    function _swapUniswap(
        address _from,
        address _to,
        uint256 _amount
    ) internal {
        require(_to != address(0));

        // Swap with uniswap
        IERC20(_from).safeApprove(currentRouter, 0);
        IERC20(_from).safeApprove(currentRouter, _amount);

        address[] memory path;

        if (_from == weth || _to == weth) {
            path = new address[](2);
            path[0] = _from;
            path[1] = _to;
        } else {
            path = new address[](3);
            path[0] = _from;
            path[1] = weth;
            path[2] = _to;
        }

        IUniswapRouterV2(currentRouter).swapExactTokensForTokens(
            _amount,
            0,
            path,
            address(this),
            now.add(60)
        );
    }

    function _swapUniswapWithPath(
        address[] memory path,
        uint256 _amount
    ) internal {
        require(path[1] != address(0));

        // Swap with uniswap
        IERC20(path[0]).safeApprove(currentRouter, 0);
        IERC20(path[0]).safeApprove(currentRouter, _amount);

        IUniswapRouterV2(currentRouter).swapExactTokensForTokens(
            _amount,
            0,
            path,
            address(this),
            now.add(60)
        );
    }

    function _swapUniswapWithPathForFeeOnTransferTokens(
        address[] memory path,
        uint256 _amount
    ) internal {
        require(path[1] != address(0));

        // Swap with uniswap
        IERC20(path[0]).safeApprove(currentRouter, 0);
        IERC20(path[0]).safeApprove(currentRouter, _amount);

        IUniswapRouterV2(currentRouter).swapExactTokensForTokensSupportingFeeOnTransferTok
            _amount,
            0,
            path,
            address(this),
            now.add(60)
        );
    }

    // **** Events **** // (forgot to put these in the live version)
    event Recovered(address indexed tokenWithdrew);
    event Notified(address indexed tokenDeposited);
}
```

## SLOC Appendix

Solidity Contracts

| Language | Files | Lines | Blanks | Comments | Code | Complex |
|---|---|---|---|---|---|---|
| Solidity | 12 | 2269 | 372 | 332 | 1565 | 263 |

Comments to Code 332/1565 = 21%

Javascript Tests

| Language | Files | Lines | Blanks | Comments | Code | Complex |
|---|---|---|---|---|---|---|
| JavaScript | 0 | 0 | 0 | 0 | 0 | 0 |

Tests to Code 0/0 = 0%