0.7

Idle v4

Score: 83%

Overview

This is an Idle Finance Process Quality Review completed on 07/10/2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Nick of DeFiSafety. Check out our Telegram.

The final score of the review is **83%**, a **PASS.** 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 blockchains used by this protocol. This report covers all of the blockchains upon which the protocol is deployed.

> **Chain:** Ethereum

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra
Celo
Arbitrum
Solana

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is here. This review will answer the following questions:

1) Are the executing code addresses readily available? (%)
2) Is the code actively being used? (%)
3) Is there a public software repository? (Y/N)
4) Is there a development history visible? (%)
5) Is the team public (not anonymous)? (Y/N)

1) Are the executing code addresses readily available? (%)

> **Answer:** 100%

They are available at website https://developers.idle.finance/contracts-and-codebase, as indicated in the Appendix.

Guidance:

100%	Clearly labelled and on website, docs or repo, quick to find
70%	Clearly labelled and on website, docs or repo but takes a bit of looking
40%	Addresses in mainnet.json, in discord or sub graph, etc
20%	Address found but labeling not clear or easy to find
0%	Executing addresses could not be found

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

> **Answer:** 70%

Activity is more that 10 transactions a week on contract *IdleDAIYield*, as indicated in the Appendix.

Guidance:

100%	More than 10 transactions a day
70%	More than 10 transactions a week
40%	More than 10 transactions a month
10%	Less than 10 transactions a month
0%	No activity

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

> **Answer:** Yes

GitHub: https://github.com/Idle-Labs/

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%

At 400 commits and 6 branches, this protocol's commitment to development history can be described as anything but idle.

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

Location: https://github.com/Idle-Labs/idle-contracts/graphs/contributors

---

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://developers.idle.finance/

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

> **Answer:** Yes

The basic software functions are documented at an identification level.

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

> ⓘ **Answer:** 50%

The software function documentation identifies the underlying functions behind the Idle Tranches contract in the README.md of the respective GitHub repository. However, none of the other Idle contracts have this detailed information apart from a architecture diagram.

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:** 63%

Code examples are in the Appendix. As per the SLOC, there is 63% 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:** 40%

The documentation lists some of the functions, but there is no explicit traceability and there is no coverage

(in software terms) of what the contracts actually are. The UI and some functions are covered, but this is less relevant.

Guidance:

100% Clear explicit traceability between code and documentation at a requirement level for all code
60% Clear association between code and documents via non explicit traceability
40% Documentation lists all the functions and describes their functions
0% No connection between documentation and code

How to improve this score:

This score can improve by adding traceability from documentation to code such that it is clear where each outlined function is coded in the source code. For reference, check the SecurEth guidelines on traceability.

Testing

This section looks at the software testing available. It is explained in this document. This section answers the following questions;

11) Full test suite (Covers all the deployed code) (%)
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)
13) Scripts and instructions to run the tests (Y/N)
14) Report of the results (%)
15) Formal Verification test done (%)
16) Stress Testing environment (%)

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

> **Answer:** 100%

Code examples are in the Appendix. As per the SLOC, there is 145% 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 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

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

> **Answer:** 50%

Idle has an average of 33% code coverage across all contracts according to their Quantstamp audits. However, since they clearly have a robust set of tests, we will up their score to 50% according to the guidance below.

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

Scripts/Instructions location: https://github.com/Idle-Labs/idle-contracts

14) Report of the results (%)

> **Answer:** 0%

There is no test report in the Idle 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%

No formal verification test has been undertaken.

16) Stress Testing environment (%)

> ✓ **Answer:** 100%

Idle is deployed in full to the Kovan testnet.

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%

Both Idle yield and Idle governance have been continuously audited before, during, and after their launches.

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:** 90%

Idle Finance's Immunefi program offers up to $500k.

Guidance:

100% Bounty is 10% TVL or at least $1M AND active program (see below)
90% Bounty is 5% TVL or at least 500k AND active program
80% Bounty is 5% TVL or at least 500k
70% Bounty is 100k or over AND active program
60% Bounty is 100k or over
50% Bounty is 50k or over AND active program
40% Bounty is 50k or over
20% Bug bounty program bounty is less than 50k
0% No bug bounty program offered

An active program means that a third party (such as Immunefi) is actively driving hackers to the site. An inactive program would be static mentions on the docs.

Access Controls

This section covers the documentation of special access controls for a DeFi protocol. The admin access controls are the contracts that allow updating contracts or coefficients in the protocol. Since these contracts can allow the protocol admins to "change the rules", complete disclosure of capabilities is vital for user's transparency. It is explained in this document. The questions this section asks are as follow;

19) Can a user clearly and quickly find the status of the admin controls?
20) Is the information clear and complete?
21) Is the information in non-technical terms that pertain to the investments?
22) Is there Pause Control documentation including records of tests?

19) Can a user clearly and quickly find the status of the access controls (%)

> **Answer:** 100%

Idle finance's admin powers section is clearly labelled in the 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:** 90%

a) All contracts are clearly labelled as upgradeable (or not) -- 30% -- IdleToken contract is clearly identified as upgradeable, and that it controls all other contracts. The Pause Guardian is also identified as upgradeable and the rebalancer is too.
b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% -- IDLE governance is the sole owner of admin privileges.
c) The capabilities for change in the contracts are described -- 30% -- capacities are clearly identified.

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%

The information is in 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:** 80%

A pause guardian is clearly identified with a detailed explanation on how it functions.

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

PQ Audit Scoring Matrix (v0.7)	Total	Idle Finance	
	Points	Answer	Points

PQ Audit Scoring Matrix (v0.7)	Points	Answer	Points
Total	260		214.65
Code and Team			**83%**
1) Are the executing code addresses readily available? (%)	20	100%	20
2) Is the code actively being used? (%)	5	70%	3.5
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	50%	7.5
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	5	63%	3.15
10) Is it possible to trace from software documentation to the implementation in code (%)	10	40%	4
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	0%	0
15) Formal Verification test done (%)	5	0%	0
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	90%	9
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	90%	9
21) Is the information in non-technical terms	10	30%	3
22) Is there Pause Control documentation including records of tests	10	80%	8
Section Scoring			
Code and Team	50	97%	
Documentation	45	66%	
Testing	50	65%	
Security	80	99%	
Access Controls	35	71%	

Executing Code Appendix

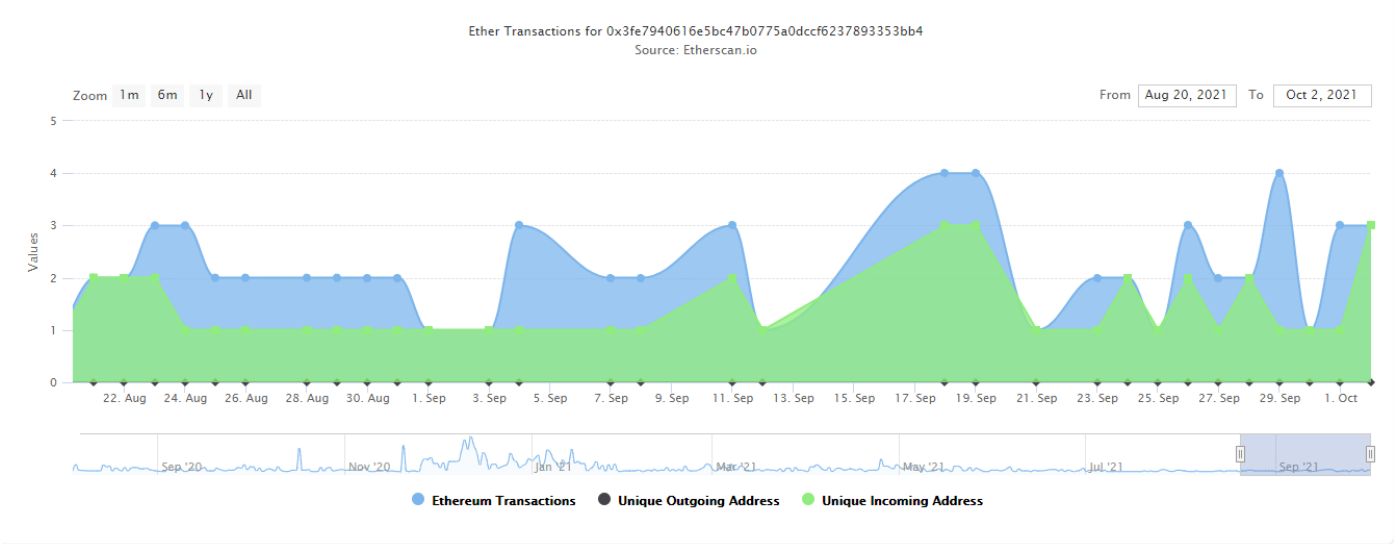
Mainnet

Idle protocol V4:

CONTRACT	STRATEGY	ABI	ADDRESS
IdleDAI	Best Yield	JSON	0x3fE7940616e5Bc47b0775a0dccf6237893353bB4
IdleUSDC	Best Yield	JSON	0x5274891bEC421B39D23760c04A6755eCB444797C
IdleUSDT	Best Yield	JSON	0xF34842d05A1c888Ca02769A633DF37177415C2f8
IdleSUSD	Best Yield	JSON	0xf52cdcd458bf455aed77751743180ec4a595fd3f

IdleTUSD	Best Yield	JSON	0xc278041fDD8249FE4c1Aad1193876857EEa3D68c
IdleWBTC	Best Yield	JSON	0x8C81121B15197fA0eEaEE1DC75533419DcfD3151
IdleWETH	Best Yield	JSON	0xC8E6CA6E96a326dC448307A5fDE90a0b21fd7f80
IdleRAI	Best Yield	JSON	0x5C960a3DCC01BE8a0f49c02A8ceBCAcf5D07fABe
IdleFEI (Beta)	Best Yield	JSON	0xb2d5CB72A621493fe83C6885E4A776279be595bC
IdleDAI	Risk Adjusted	JSON	0xa14eA0E11121e6E951E87c66AFe460A00BCD6A16
IdleUSDC	Risk Adjusted	JSON	0x3391bc034f2935ef0e1e41619445f998b2680d35
IdleUSDT	Risk Adjusted	JSON	0x28fAc5334C9f7262b3A3Fe707e250E01053e07b5

Code Used Appendix



Example Code Appendix

```
1  /**
2   * @title: Idle Token Governance main contract
3   * @summary: ERC20 that holds pooled user funds together
4   *           Each token rapresent a share of the underlying pools
5   *           and with each token user have the right to redeem a portion of these pools
6   * @author: Idle Labs Inc., idle.finance
7   */
8  pragma solidity 0.5.16;
9  import "@openzeppelin/contracts-ethereum-package/contracts/math/SafeMath.sol";
10 import "@openzeppelin/contracts-ethereum-package/contracts/token/ERC20/SafeERC20.sol";
11
12 import "@openzeppelin/contracts-ethereum-package/contracts/token/ERC20/ERC20.sol";
13 import "@openzeppelin/contracts-ethereum-package/contracts/token/ERC20/ERC20Detailed.sol";
14
15
```

```
import "@openzeppelin/contracts-ethereum-package/contracts/utils/ReentrancyGuard.sol";
import "@openzeppelin/contracts-ethereum-package/contracts/ownership/Ownable.sol";
import "@openzeppelin/contracts-ethereum-package/contracts/lifecycle/Pausable.sol";

import "@openzeppelin/upgrades/contracts/Initializable.sol";

import "./interfaces/iERC20Fulcrum.sol";
import "./interfaces/ILendingProtocol.sol";
import "./interfaces/IGovToken.sol";
import "./interfaces/IIdleTokenV3_1.sol";
import "./interfaces/IERC3156FlashBorrower.sol";
import "./interfaces/IAaveIncentivesController.sol";

import "./interfaces/Comptroller.sol";
import "./interfaces/CERC20.sol";
import "./interfaces/AToken.sol";
import "./interfaces/IdleController.sol";
import "./interfaces/IIdleTokenHelper.sol";

import "./GST2ConsumerV2.sol";

contract IdleTokenGovernance is Initializable, ERC20, ERC20Detailed, ReentrancyGuard, Ownal
  using SafeERC20 for IERC20;
  using SafeMath for uint256;

  uint256 private constant ONE_18 = 10**18;
  // State variables
  // eg. DAI address
  address public token;
  // eg. iDAI address
  address private iToken;
  // eg. cDAI address
  address private cToken;
  // Idle rebalancer current implementation address
  address public rebalancer;
  // Address collecting underlying fees
  address public feeAddress;
  // Last iToken price, used to pause contract in case of a black swan event
  uint256 public lastITokenPrice;
  // eg. 18 for DAI
  uint256 private tokenDecimals;
  // Max unlent assets percentage for gas friendly swaps
  uint256 public maxUnlentPerc; // 100000 == 100% -> 1000 == 1%
  // Current fee on interest gained
  uint256 public fee;
  // eg. [cTokenAddress, iTokenAddress, ...]
  address[] public allAvailableTokens;
  // eg. [COMPAddress, CRVAddress, ...]
  address[] public govTokens;
  // last fully applied allocations (ie when all liquidity has been correctly placed)
  // eg. [5000, 0, 5000, 0] for 50% in compound, 0% fulcrum, 50% aave, 0 dydx. same order
  uint256[] public lastAllocations;
  // Map that saves avg idleToken price paid for each user, used to calculate earnings
```

```
68
    mapping(address => uint256) public userAvgPrices;
69  // eg. cTokenAddress => IdleCompoundAddress
70  mapping(address => address) public protocolWrappers;
71  // array with last balance recorded for each gov tokens
72  mapping (address => uint256) public govTokensLastBalances;
73  // govToken -> user_address -> user_index eg. usersGovTokensIndexes[govTokens[0]][msg.se
74  mapping (address => mapping (address => uint256)) public usersGovTokensIndexes;
75  // global indices for each gov tokens used as a reference to calculate a fair share for
76  mapping (address => uint256) public govTokensIndexes;
77  // Map that saves amount with no fee for each user
78  mapping(address => uint256) private userNoFeeQty;
79  // variable used for avoid the call of mint and redeem in the same tx
80  bytes32 private _minterBlock;
81
82  // Events
83  event Rebalance(address _rebalancer, uint256 _amount);
84  event Referral(uint256 _amount, address _ref);
85
86  // ########## IdleToken V4_1 updates
87  // Idle governance token
88  address public constant IDLE = address(0x875773784Af8135eA0ef43b5a374AaD105c5D39e);
89  // Compound governance token
90  address public constant COMP = address(0xc00e94Cb662C3520282E6f5717214004A7f26888);
91  uint256 private constant FULL_ALLOC = 100000;
92
93  // Idle distribution controller
94  address public constant idleController = address(0x275DA8e61ea8E02d51EDd8d0DC5c0E62b4CDB
95  // oracle used for calculating the avgAPR with gov tokens
96  address public oracle;
97  // eg cDAI -> COMP
98  mapping(address => address) private protocolTokenToGov;
99  // Whether openRebalance is enabled or not
100 bool public isRiskAdjusted;
101 // last allocations submitted by rebalancer
102 uint256[] private lastRebalancerAllocations;
103
104 // ########## IdleToken V5 updates
105 // Fee for flash loan
106 uint256 public flashLoanFee;
107 // IdleToken helper address
108 address public tokenHelper;
109
110 /**
111 * @dev Emitted on flashLoan()
112 * @param target The address of the flash loan receiver contract
113 * @param initiator The address initiating the flash loan
114 * @param amount The amount flash borrowed
115 * @param premium The flash loan fee
116 **/
117 event FlashLoan(
118   address indexed target,
119   address indexed initiator,
```

```
    uint256 amount,
    uint256 premium
  );

  // Addresses for stkAAVE distribution from Aave
  address public constant stkAAVE = address(0x4da27a545c0c5B758a6BA100e3a049001de870f5);
  address private aToken;
  // ########## End IdleToken V5 updates

  // ERROR MESSAGES:
  // 0 = is 0
  // 1 = already initialized
  // 2 = length is different
  // 3 = Not greater then
  // 4 = lt
  // 5 = too high
  // 6 = not authorized
  // 7 = not equal
  // 8 = error on flash loan execution
  // 9 = Reentrancy

  function _init() public {
    require(oracle == 0xB5A8f07dD4c3D315869405d702ee8F6EA695E8C5);
    oracle = 0x758C10272A15f0E9D50Cbc035ff9a046945da0F2;
    flashLoanFee = 20;
  }

  // onlyOwner
  /**
   * It allows owner to modify allAvailableTokens array in case of emergency
   * ie if a bug on a interest bearing token is discovered and reset protocolWrappers
   * associated with those tokens.
   *
   * @param protocolTokens : array of protocolTokens addresses (eg [cDAI, iDAI, ...])
   * @param wrappers : array of wrapper addresses (eg [IdleCompound, IdleFulcrum, ...])
   * @param _newGovTokens : array of governance token addresses
   * @param _newGovTokensEqualLen : array of governance token addresses for each
   *  protocolToken (addr0 should be used for protocols with no govToken)
   */
  function setAllAvailableTokensAndWrappers(
    address[] calldata protocolTokens,
    address[] calldata wrappers,
    address[] calldata _newGovTokens,
    address[] calldata _newGovTokensEqualLen
  ) external onlyOwner {
    require(protocolTokens.length == wrappers.length, "2");
    require(_newGovTokensEqualLen.length >= protocolTokens.length, '3');

    govTokens = _newGovTokens;

    address newGov;
    address protToken;
    for (uint256 i = 0; i < protocolTokens.length; i++) {
      protToken = protocolTokens[i];
```

```
      protToken = protocolTokens[i];
      require(protToken != address(0) && wrappers[i] != address(0), "0");
      protocolWrappers[protToken] = wrappers[i];

      // set protocol token to gov token mapping
      newGov = _newGovTokensEqualLen[i];
      if (newGov != IDLE) {
        protocolTokenToGov[protToken] = newGov;
      }
    }

    allAvailableTokens = protocolTokens;
  }

  /**
   * It allows owner to set the cToken address
   *
   * @param _cToken : new cToken address
   */
  function setCToken(address _cToken)
    external onlyOwner {
      require((cToken = _cToken) != address(0), "0");
  }

  /**
   * It allows owner to set the aToken address
   *
   * @param _aToken : new aToken address
   */
  function setAToken(address _aToken)
    external onlyOwner {
      require((aToken = _aToken) != address(0), "0");
  }

  /**
   * It allows owner to set the tokenHelper address
   *
   * @param _tokenHelper : new tokenHelper address
   */
  function setTokenHelper(address _tokenHelper)
    external onlyOwner {
      require((tokenHelper = _tokenHelper) != address(0), "0");
  }

  /**
   * It allows owner to set the IdleRebalancerV3_1 address
   *
   * @param _rebalancer : new IdleRebalancerV3_1 address
   */
  function setRebalancer(address _rebalancer)
    external onlyOwner {
      require((rebalancer = _rebalancer) != address(0), "0");
  }
```

```
226
227  /**
228   * It allows owner to set the fee (1000 == 10% of gained interest)
229   *
230   * @param _fee : fee amount where 100000 is 100%, max settable is 10%
231   */
232  function setFee(uint256 _fee)
233    external onlyOwner {
234      // 100000 == 100% -> 10000 == 10%
235      require((fee = _fee) <= FULL_ALLOC / 10, "5");
236  }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	12	2272	238	789	1245	197

Comments to Code 789/1245 = 63%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	4	3096	382	9904	1810	11

Tests to Code 1810/1245 = 145%