0.7

Instadapp v2 0.7 UPDATE Process Quality Review

Score: 72%

Overview

This is an Instadapp Process Quality Review completed on 21/09/2021. 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 **72%**, 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 blockchain used by this protocol.

> **Chain:** Ethereum, Arbitrum, Polygon

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra
Celo
Arbitrum

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://docs.instadapp.io/networks/mainnet, 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:** 100%

Activity is 12 transactions a day on contract 0x2971AdFa57b20E5a416aE5a708A8655A9c74f723, 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/instadapp.

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%

There are 431 commits and 4 branches, making Instadapp's repository healthy.

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/Instadapp/smart-contract/graphs/contributors.

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

Location: https://docs.instadapp.io/.

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

> **Answer:** Yes

The basic software functions are covered by the 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:** 80%

Instadapp documents the software functions of all their major mainnet contracts, except for the Implementations contracts in addition to the TimeLock and Governance contracts.

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

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 17% 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%

Instadapp lists all of their functions and describes their use cases without providing a visual representation of their implementation within the protocol's source code. Although they do this for their SDK guide amongst others, they do not do this for their main smart contracts.

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:** 0%

Code examples are in the Appendix. As per the SLOC, there is 3% 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

How to improve this score:

This score can improved 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:** 30%

An OpenZeppelin audit mentions "sparse" code coverage, which indicates tests have been run, but given the TtC value these tests are incomplete.

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/InstaDApp/smart-contract

14) Report of the results (%)

> **Answer:** 0%

No test report was found.

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 could be found.

16) Stress Testing environment (%)

> ✓ **Answer:** 100%

Evidence of testing on the Kovan testnet could be found.

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%

PeckShield and Samczsun have conducted a total of 3 audits over the past 18 months.

Although only one of these audits was performed before the Instadapp v2 mainnet launch, the persistent publishing of pre-launch smart contract audit reports proves that the protocol rigorously checks their code before release at any occasion, hence the 100% score for this metric.

Notes On v2 Audit Report:

Instadapp has confirmed but not resolved the low-severity issue of lack of sanity checks highlighted by the PeckShield team. A heightened frequency of these checks would be needed in order to properly validate the argument length of the *updateConnectors()* function within the InstaConnectorsV2 contract. Not regularly performing the needed sanity checks could lead to outdated internal mapping of the connectors which could result in lack of optimization and potential bugs.

PeckShield has underlined a medium-risk issue where Instadapp has a weakness in their master contract where connectors are added with the potential to execute the code of users' smart accounts via the delegateCall() function. This would effectively allow the access and management of user asserts. However, Instadapp has said that they would implement PeckShield's recommendation of governing this master contract via a DAO. However, there is no clear evidence that this has actually been implemented within the smart contracts' code.

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:** 50%

Instadapp's bug bounty program rewards up to $50,000 and is active.

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:** 40%

The OpenZeppelin audit contains information detailing the access controls, but this was not mentioned in the docs. There is also additional DAO information in their blog article about governance.

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% -- the docs detail which contracts are upgradeable.

b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% -- the "MicroDAO" has been given partial ownership with a roadmap towards full decentralization.

c) The capabilities for change in the contracts are described -- 30% -- contract upgradeability will be decided by the DAO.

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:** 90%

The description of the access controls is very well explained in Instadapp's blog article about their DAO-based governance. Users are clearly updated on the fact that they now govern the protocol, and why their funds are safer this way.

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%

Pause control information could not be found.

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

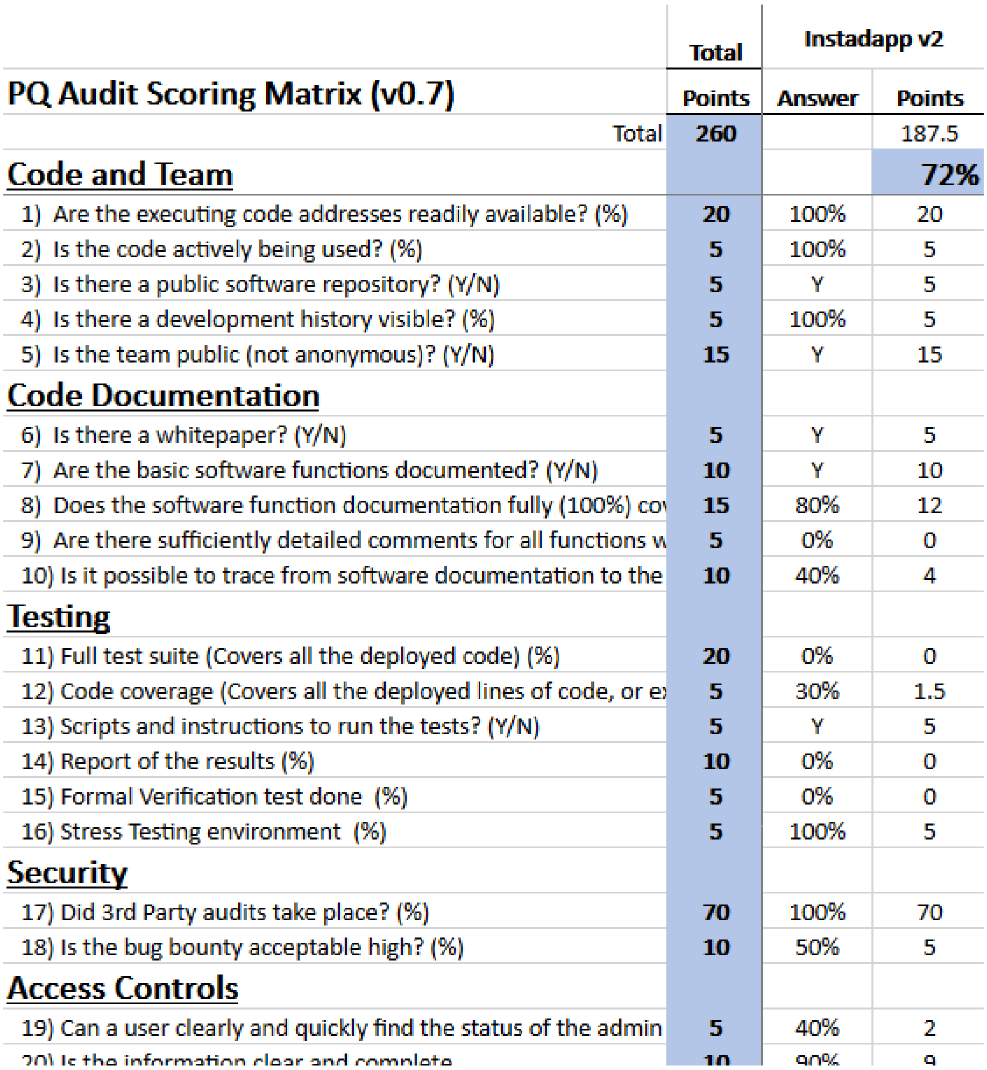
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

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.

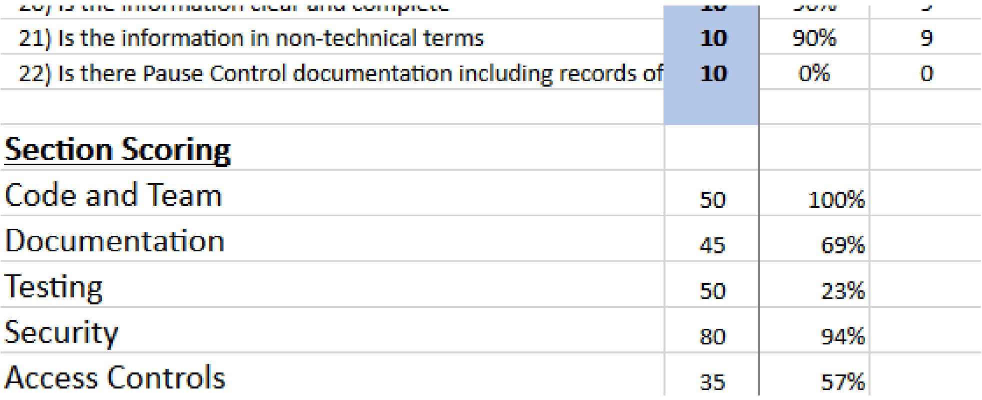
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

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

Scoring Appendix

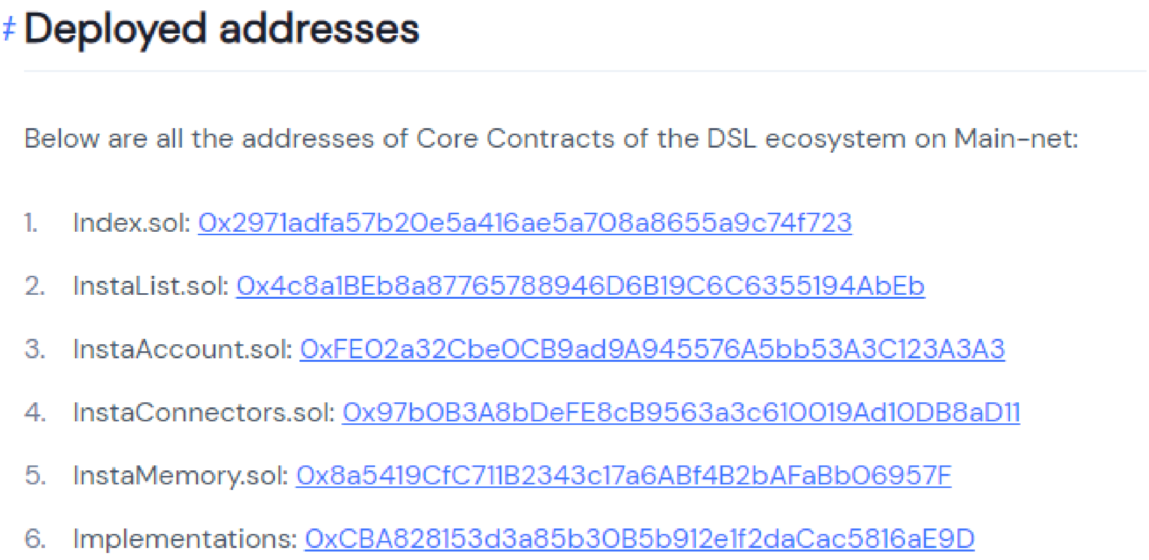


	Total	Instadapp v2	
PQ Audit Scoring Matrix (v0.7)	Points	Answer	Points
Total	260		187.5
Code and Team			72%
1) Are the executing code addresses readily available? (%)	20	100%	20
2) Is the code actively being used? (%)	5	100%	5
3) Is there a public software repository? (Y/N)	5	Y	5
4) Is there a development history visible? (%)	5	100%	5
5) Is the team public (not anonymous)? (Y/N)	15	Y	15
Code Documentation			
6) Is there a whitepaper? (Y/N)	5	Y	5
7) Are the basic software functions documented? (Y/N)	10	Y	10
8) Does the software function documentation fully (100%) cov	15	80%	12
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	40%	4
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	0%	0
12) Code coverage (Covers all the deployed lines of code, or ex	5	30%	1.5
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	50%	5
Access Controls			
19) Can a user clearly and quickly find the status of the admin	5	40%	2
20) Is the information clear and complete	10	90%	9



20) Is the information clear and complete	10	90%	9
21) Is the information in non-technical terms	10	90%	9
22) Is there Pause Control documentation including records of	10	0%	0
Section Scoring			
Code and Team	50	100%	
Documentation	45	69%	
Testing	50	23%	
Security	80	94%	
Access Controls	35	57%	

Executing Code Appendix

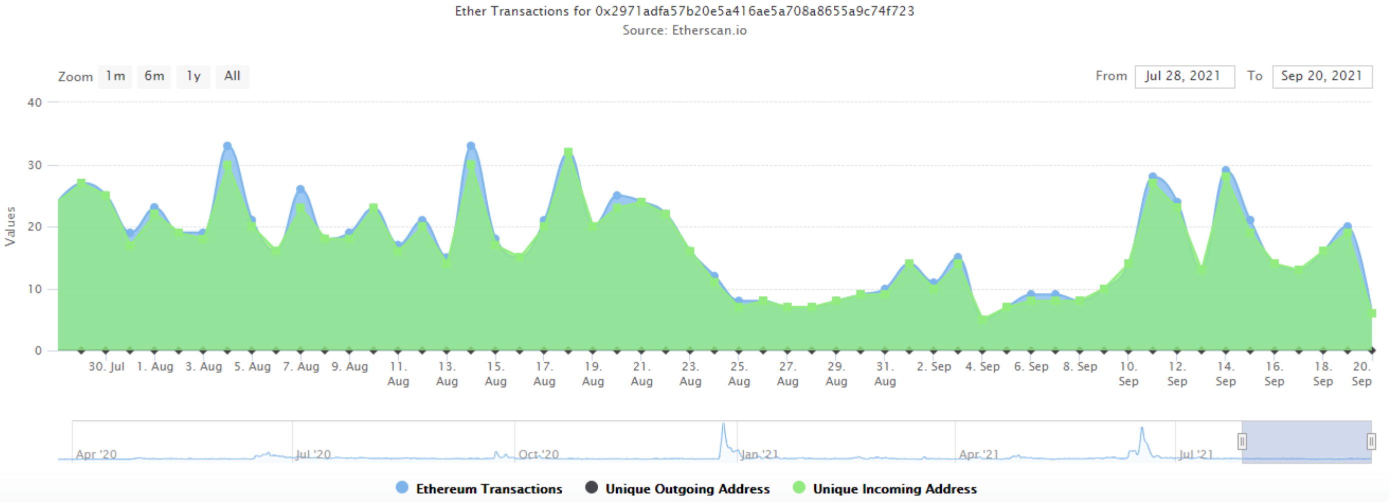


Deployed addresses

Below are all the addresses of Core Contracts of the DSL ecosystem on Main-net:

1. Index.sol: 0x2971adfa57b20e5a416ae5a708a8655a9c74f723
2. InstaList.sol: 0x4c8a1BEb8a87765788946D6B19C6C6355194AbEb
3. InstaAccount.sol: 0xFE02a32Cbe0CB9ad9A945576A5bb53A3C123A3A3
4. InstaConnectors.sol: 0x97b0B3A8bDeFE8cB9563a3c610019Ad10DB8aD11
5. InstaMemory.sol: 0x8a5419CfC711B2343c17a6ABf4B2bAFaBb06957F
6. Implementations: 0xCBA828153d3a85b30B5b912e1f2daCac5816aE9D

Code Used Appendix



Ether Transactions for 0x2971adfa57b20e5a416ae5a708a8655a9c74f723
Source: Etherscan.io

Example Code Appendix

```
pragma solidity ^0.7.0;

pragma experimental ABIEncoderV2;

/**
 * @title InstaAccount.
 * @dev DeFi Smart Account Wallet.
 */

interface IndexInterface {
    function connectors(uint version) external view returns (address);
    function check(uint version) external view returns (address);
    function list() external view returns (address);
}

interface ConnectorsInterface {
    function isConnector(address[] calldata logicAddr) external view returns (bool);
    function isStaticConnector(address[] calldata logicAddr) external view returns (bool);
}

interface CheckInterface {
    function isOk() external view returns (bool);
}

interface ListInterface {
    function addAuth(address user) external;
    function removeAuth(address user) external;
}


contract Record {

    event LogEnable(address indexed user);
    event LogDisable(address indexed user);
    event LogSwitchShield(bool _shield);

    // InstaIndex Address.
    address public immutable instaIndex;
    // The Account Module Version.
    uint public constant version = 1;
    // Auth Module(Address of Auth => bool).
    mapping (address => bool) private auth;
    // Is shield true/false.
    bool public shield;

    constructor (address _instaIndex) {
        instaIndex = _instaIndex;
    }

    /**
     * @dev Check for Auth if enabled.
     * @param user address/user/owner.
     */
    function isAuth(address user) public view returns (bool) {
        return auth[user];
    }

    /**
     * @dev Change Shield State.
    */
    function switchShield(bool _shield) external {
        require(auth[msg.sender], "not-self");
        require(shield != _shield, "shield is set");
        shield = _shield;
        emit LogSwitchShield(shield);
    }

    /**
     * @dev Enable New User.
     * @param user Owner of the Smart Account.
    */
    function enable(address user) public {
        require(msg.sender == address(this) || msg.sender == instaIndex, "not-self-index")
        require(user != address(0), "not-valid");
        require(!auth[user], "already-enabled");
        auth[user] = true;
        ListInterface(IndexInterface(instaIndex).list()).addAuth(user);
        emit LogEnable(user);
    }

    /**
     * @dev Disable User.
     * @param user Owner of the Smart Account.
    */
    function disable(address user) public {
        require(msg.sender == address(this), "not-self");
        require(user != address(0), "not-valid");
        require(auth[user], "already-disabled");
        delete auth[user];
        ListInterface(IndexInterface(instaIndex).list()).removeAuth(user);
        emit LogDisable(user);
    }

}

contract InstaAccount is Record {

    constructor (address _instaIndex) public Record(_instaIndex) {
    }

    event LogCast(address indexed origin, address indexed sender, uint value);

    receive() external payable {}

     /**
     * @dev Delegate the calls to Connector And this function is ran by cast().
     * @param _target Target to of Connector.
     * @param _data CallData of function in Connector.
     * @param _data CallData of function in Connector.

    */
    function spell(address _target, bytes memory _data) internal {
        require(_target != address(0), "target-invalid");
        assembly {
            let succeeded := delegatecall(gas(), _target, add(_data, 0x20), mload(_data), 0

            switch iszero(succeeded)
                case 1 {
                    // throw if delegatecall failed
                    let size := returndatasize()
                    returndatacopy(0x00, 0x00, size)
                    revert(0x00, size)
                }
        }
    }

    /**
     * @dev This is the main function, Where all the different functions are called
     * from Smart Account.
     * @param _targets Array of Target(s) to of Connector.
     * @param _datas Array of Calldata(S) of function.
    */
    function cast(
        address[] calldata _targets,
        bytes[] calldata _datas,
        address _origin
    )
    external
    payable
    {
        require(isAuth(msg.sender) || msg.sender == instaIndex, "permission-denied");
        require(_targets.length == _datas.length , "array-length-invalid");
        IndexInterface indexContract = IndexInterface(instaIndex);
        bool isShield = shield;
        if (!isShield) {
            require(ConnectorsInterface(indexContract.connectors(version)).isConnector(_ta
        } else {
            require(ConnectorsInterface(indexContract.connectors(version)).isStaticConnect
        }
        for (uint i = 0; i < _targets.length; i++) {
            spell(_targets[i], _datas[i]);
        }
        address _check = indexContract.check(version);
        if (_check != address(0) && !isShield) require(CheckInterface(_check).isOk(), "not-
        emit LogCast(_origin, msg.sender, msg.value);
    }

}
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	319	9309	1529	1120	6660	440

Comments to Code 1120/6660 = 17%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	4	280	38	35	207	17

Tests to Code 207/6660 = 3%