mStable Process Quality Review

Score: 90%

Overview

This is a mStable Process Quality Review completed on July 21st 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 90%, an excellent 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

Chain

This section indicates the blockchain used by this protocol.

> ✓ **Chain:** Ethereum, Polygon

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche

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 , 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 10 transactions a day on contract *IncentivisedVotingLockup.sol*, 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/mstable.

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 574 and 5 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

Location: https://docs.mstable.org/appendix/about-us.

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.mstable.org/

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

✓ **Answer:** Yes

The basic software functions (code) of the mStable infrastructures and assets are well-documented.

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

ⓘ **Answer: 7**0%

There is not software function documentation, but very thorough and technical capabilities that mention the main contracts. This gives a score of 70%. The documented software functions (code) of mStable cover their app and its functions to their protocol architecture, as well as their data processing and validation through mStable-js.

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:** 90%

Code examples are in the Appendix. As per the SLOC, there is 58% commenting to code (CtC). The commenting follows NatSpec fully for that reason the score for commenting is 90%

The Comments to Code (CtC) ratio is the primary metric for this score.

Note: The CtC was calculated using only files that were authored by the mStable developers. This means that we did not include any interface, OpenZeppelin, and mock files (mock files were excluded because they are, well, mocks that serve no executive purpose at the moment).

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%

With no explicit software documentation, there cannot be any traceability.

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

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 4097% 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:** 100%

mStable has a 96% coveralls code coverage score for their main contracts. However, they also have a 100% ConsenSys Diligence code coverage score from their audit report.

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

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

> ✓ **Answer:** Yes

Scrips/Instructions location: https://github.com/mstable/mStable-contracts/blob/master-v2/README.md.

14) Report of the results (%)

> ✓ **Answer:** 100%

Detailed test report from coveralls, as well as passing CI reports from the mStable's GitHub repository.

Guidance:

100%	Detailed test report as described below
70%	GitHub code coverage report visible
0%	No test report evident

15) Formal Verification test done (%)

⚠ **Answer:** 0%

No evidence of a mStable Formal Verification was found in their documentation or in web searches.

16) Stress Testing environment (%)

✓ **Answer:** 100%

There is clear evidence of mStable's test-net smart contract usages in their contracts' documentation.

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%

mStable has had audits from ConsenSys Diligence and Bramah Systems (before deployment), as well as from Certik and PeckShield (after deployment). All audit reports can be found here.

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: 70%

mStable has a Immunefi Bug Bounty Program that is live and offers as much as 100k for the most critical of findings.

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%

Governance can easily be found in the Governance section of their 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) Most of the contracts are immutable, and few are upgradeable. This is described here.

b) There are defined roles in the governance section of the mStable documentation.

c) The capabilities for change in contracts through voting are described here.

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%

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

Answer: 90%

All information pertaining governance and safety are all described in very user-friendly terms.

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

22) Is there Pause Control documentation including records of tests (%)

Answer: 80%

Pause Control is mentioned in "Areas of interest", and recent governance tests are recorded here.

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

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

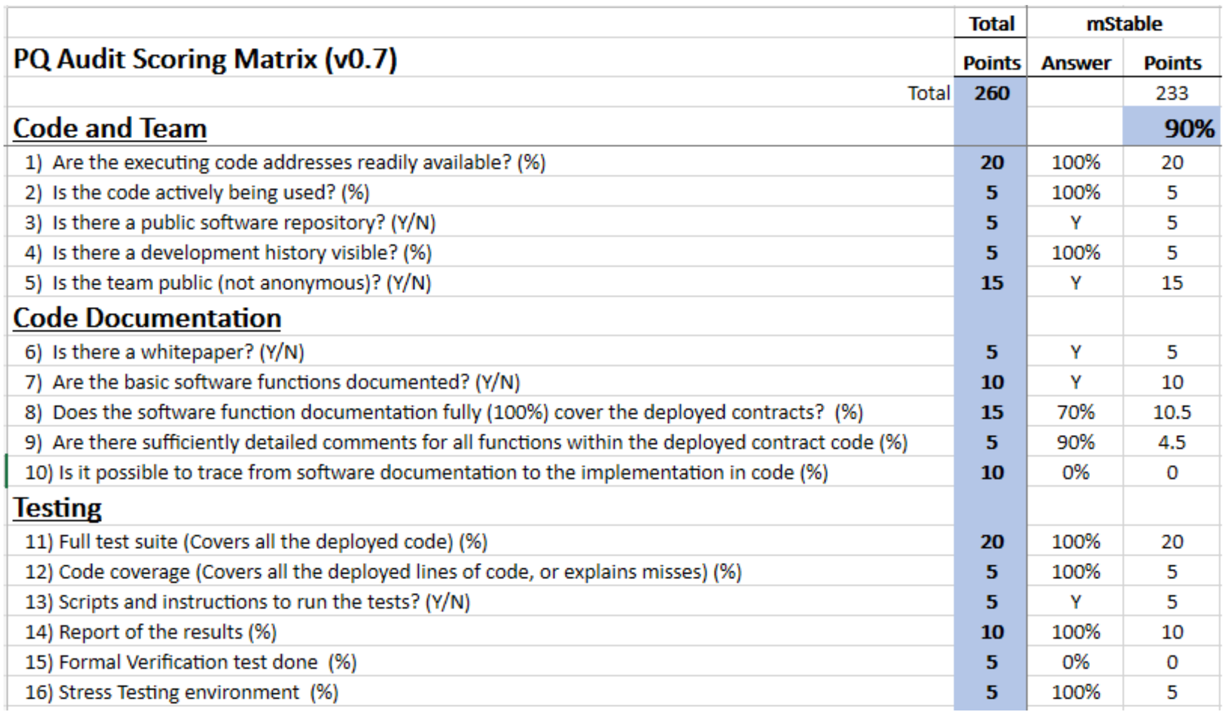
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

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.

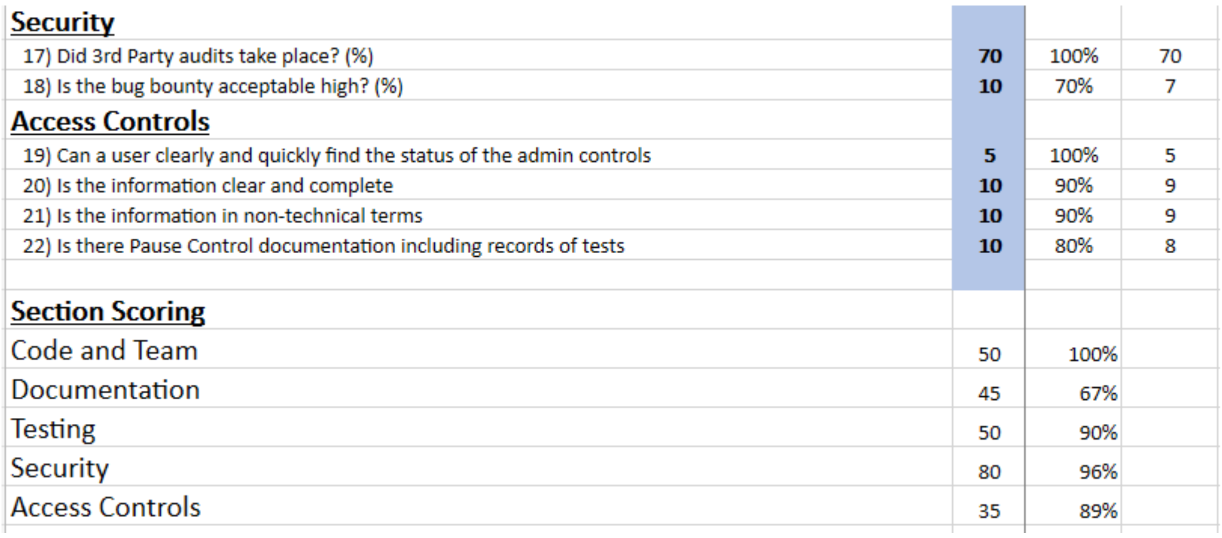
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

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

Scoring Appendix

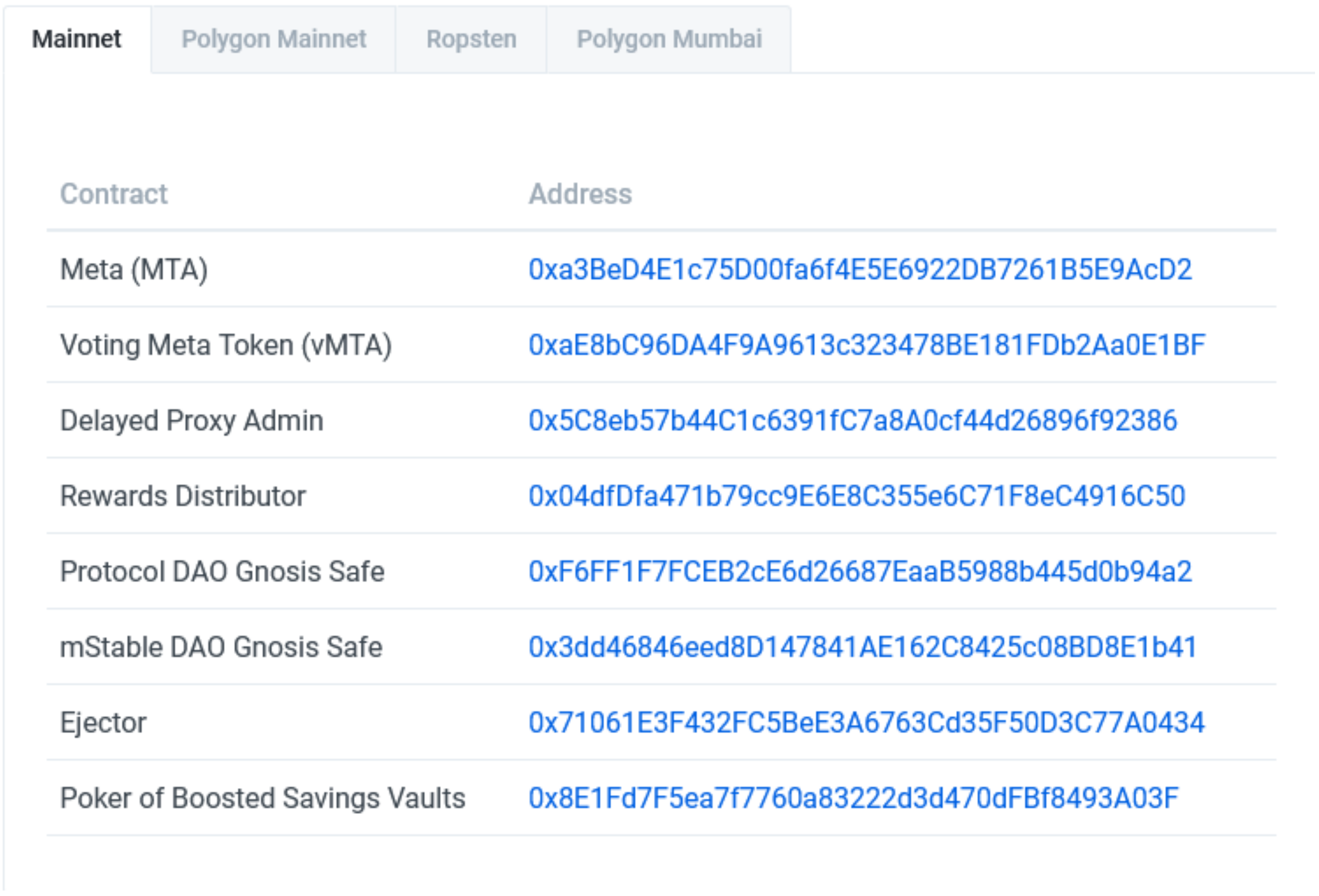


PQ Audit Scoring Matrix (v0.7)	Total Points	mStable Answer	mStable Points
Total	260		233
Code and Team			**90%**
1) Are the executing code addresses readily available? (%)	20	100%	20
2) Is the code actively being used? (%)	5	100%	5
3) Is there a public software repository? (Y/N)	5	Y	5
4) Is there a development 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%) cover the deployed contracts? (%)	15	70%	10.5
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	5	90%	4.5
10) Is it possible to trace from software documentation to the implementation in code (%)	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 explains misses) (%)	5	100%	5
13) Scripts and instructions to run the tests? (Y/N)	5	Y	5
14) Report of the results (%)	10	100%	10
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**	70%	7
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**	90%	9
22) Is there Pause Control documentation including records of tests	**10**	80%	8
Section Scoring			
Code and Team	50	100%	
Documentation	45	67%	
Testing	50	90%	
Security	80	96%	
Access Controls	35	89%	

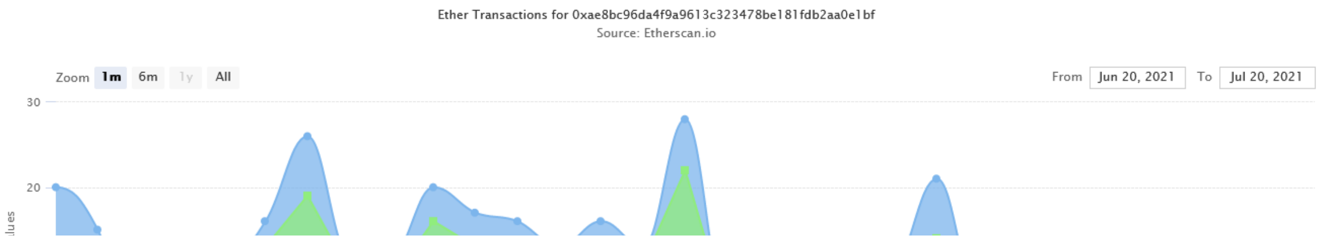
Executing Code Appendix

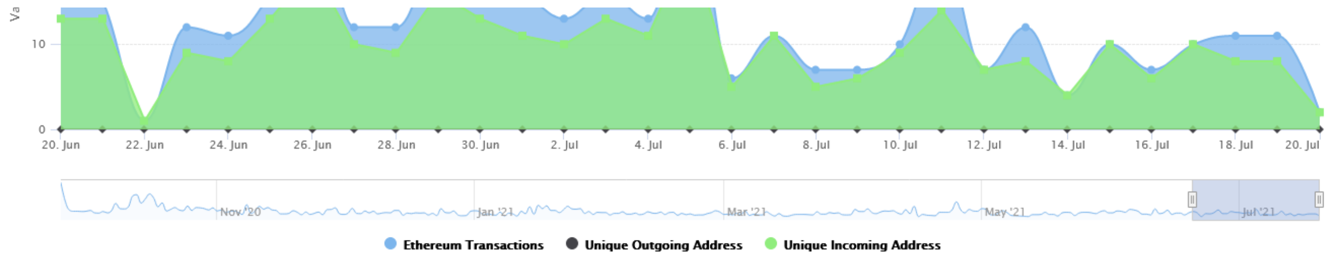


Mainnet | Polygon Mainnet | Ropsten | Polygon Mumbai

Contract	Address
Meta (MTA)	0xa3BeD4E1c75D00fa6f4E5E6922DB7261B5E9AcD2
Voting Meta Token (vMTA)	0xaE8bC96DA4F9A9613c323478BE181FDb2Aa0E1BF
Delayed Proxy Admin	0x5C8eb57b44C1c6391fC7a8A0cf44d26896f92386
Rewards Distributor	0x04dfDfa471b79cc9E6E8C355e6C71F8eC4916C50
Protocol DAO Gnosis Safe	0xF6FF1F7FCEB2cE6d26687EaaB5988b445d0b94a2
mStable DAO Gnosis Safe	0x3dd46846eed8D147841AE162C8425c08BD8E1b41
Ejector	0x71061E3F432FC5BeE3A6763Cd35F50D3C77A0434
Poker of Boosted Savings Vaults	0x8E1Fd7F5ea7f7760a83222d3d470dFBf8493A03F

Code Used Appendix





Example Code Appendix

```
/**
 * @title   Nexus
 * @author  mStable
 * @notice  Address provider and system kernel, also facilitates governan
 * @dev     The Nexus is mStable's Kernel, and allows the publishing and 
 *          of new system Modules. Other Modules will read from the Nexus
 *          VERSION: 3.0
 *          DATE:    2021-04-15
 */
contract Nexus is INexus, DelayedClaimableGovernor {
    event ModuleProposed(bytes32 indexed key, address addr, uint256 times
    event ModuleAdded(bytes32 indexed key, address addr, bool isLocked);
    event ModuleCancelled(bytes32 indexed key);
    event ModuleLockRequested(bytes32 indexed key, uint256 timestamp);
    event ModuleLockEnabled(bytes32 indexed key);
    event ModuleLockCancelled(bytes32 indexed key);

    /** @dev Struct to store information about current modules */
    struct Module {
        address addr; // Module address
        bool isLocked; // Module lock status
    }

    /** @dev Struct to store information about proposed modules */
    struct Proposal {
        address newAddress; // Proposed Module address
        uint256 timestamp; // Timestamp when module upgrade was proposed
    }

    // 1 week delayed upgrade period
    uint256 public constant UPGRADE_DELAY = 1 weeks;

    // Module-key => Module
    mapping(bytes32 => Module) public modules;
    // Module-address => Module-key
    mapping(address => bytes32) private addressToModule;
    // Module-key => Proposal
    mapping(bytes32 => Proposal) public proposedModules;
    // Module-key => Timestamp when lock was proposed
    mapping(bytes32 => uint256) public proposedLockModules;

    // Init flag to allow add modules at the time of deplyment without del
    bool public initialized = false;

    /**
     * @dev Modifier allows functions calls only when contract is not init
     */
    modifier whenNotInitialized() {
        require(!initialized, "Nexus is already initialized");
        _;
    }

    /**
     * @dev Initialises the Nexus and adds the core data to the Kernel (i
     * @param _governorAddr Governor address
     */
    constructor(address _governorAddr) DelayedClaimableGovernor(_governor

    // FIXME can this function be avoided as it just calls the super func
    function governor() public view override(Governable, INexus) returns
        return super.governor();
    }

    /**
     * @dev Adds multiple new modules to the system to initialize the
     *      Nexus contract with default modules. This should be called fir
     *      after deploying Nexus contract.
     * @param _keys         Keys of the new modules in bytes32 form
     * @param _addresses    Contract addresses of the new modules
     * @param _isLocked     IsLocked flag for the new modules
     * @param _governorAddr New Governor address
     * @return bool         Success of publishing new Modules
     */
    function initialize(
        bytes32[] calldata _keys,
        address[] calldata _addresses,
        bool[] calldata _isLocked,
        address _governorAddr
    ) external onlyGovernor whenNotInitialized returns (bool) {
        uint256 len = _keys.length;
        require(len > 0, "No keys provided");
        require(len == _addresses.length, "Insufficient address data");
        require(len == _isLocked.length, "Insufficient locked statuses");

        for (uint256 i = 0; i < len; i++) {
            _publishModule(_keys[i], _addresses[i], _isLocked[i]);
        }

        if (_governorAddr != governor()) _changeGovernor(_governorAddr);

        initialized = true;
        return true;
    }

    /***************************************
                MODULE ADDING
    ****************************************/

    /**
     * @dev Propose a new or update existing module
     * @param _key  Key of the module
     * @param _addr Address of the module
     */
    function proposeModule(bytes32 _key, address _addr) external override
        require(_key != bytes32(0x0), "Key must not be zero");
        require(_addr != address(0), "Module address must not be 0");
        require(!modules[_key].isLocked, "Module must be unlocked");
        require(modules[_key].addr != _addr, "Module already has same add
        Proposal storage p = proposedModules[_key];
        require(p.timestamp == 0, "Module already proposed");

        p.newAddress = _addr;
        p.timestamp = block.timestamp;
        emit ModuleProposed(_key, _addr, block.timestamp);
    }

    /**
     * @dev Cancel a proposed module request
     * @param _key Key of the module
     */
    function cancelProposedModule(bytes32 _key) external override onlyGov
        uint256 timestamp = proposedModules[_key].timestamp;
        require(timestamp > 0, "Proposed module not found");

        delete proposedModules[_key];
        emit ModuleCancelled(_key);
    }

    /**
     * @dev Accept and publish an already proposed module
     * @param _key Key of the module
     */
    function acceptProposedModule(bytes32 _key) external override onlyGov
        _acceptProposedModule(_key);
    }

    /**
     * @dev Accept and publish already proposed modules
     * @param _keys Keys array of the modules
     */
    function acceptProposedModules(bytes32[] calldata _keys) external ove
        uint256 len = _keys.length;
        require(len > 0, "Keys array empty");
```

```
144
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	49	11939	1334	3893	6712	680

Comments to Code 3893/6712 = 58%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
TypeScript	59	23163	2298	2092	18773	814
JSON	18	256276	0	0	256276	0

Tests to Code 275049/6712 = 4097%