

0.7

MakerDAO 0.7 Process Quality Review

Score: 81%

Overview

This is a [MakerDAO](#) Process Quality Review completed on 14/10/2021. 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 81%, 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.

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

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, Binance Smart Chain, Polygon, Avalanche, Arbitrum,

Harmony, Moonriver, Fantom, Klaytn, xDAI and Sora

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:** 20%

They are available at website https://github.com/makerdao/spells-mainnet/blob/master/src/test/addresses_mainnet.sol, as indicated in the [Appendix](#). The documents contain

no mention of non-token contract addresses and digging through 250 repositories yielded few results.

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 more than 10 transactions a day on contract [Maker: MCD Join DAI](#), 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/makerdao>

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 251 repositories with some containing in excess of 800 commits, Maker's development history can be tolerated.

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/makerdao/mcd-changelog/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.makerdao.com/>

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



Answer: Yes

Basic software functions are [documented](#).

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



Answer: 100%

Maker's documentation is impressive and covers all deployed contracts in a [highly organised](#) fashion.

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: 36%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 36% 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 clear association from the documents to the code, but there is no explicit traceability to the implementation.

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 222% 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%

Code coverage is mentioned, but [proof could not be found](#). Nevertheless, there's evidently heavy testing on this protocol.

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 be 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/makerdao/spells-mainnet#test>

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:** 100%

Although Maker says that they have undergone [Formal Verification](#) testing, there exists no link to a report of any kind. However we will score it as complete.

16) Stress Testing environment (%)

 **Answer:** 100%

Maker uses [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%

Maker has undergone multiple audits for many of its products, all of which were conducted shortly before or after [contract deployment](#).

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: 70%

Maker offers a \$100K [HackerOne bug bounty](#) program.

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%

Maker's Governance and access controls documentations are [clearly outlined in their docs](#). In addition, there is also in-depth

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% -- contracts are labelled as upgradeable.
- b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% -- governance is indicated as the owner.
- c) The capabilities for change in the contracts are described -- 30% -- there is detailed information as to what can change in these contracts.

Guidance:

All the contracts are immutable -- 100% OR

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

How to improve this score:

Create a document that covers the items described above. An [example](#) is enclosed.

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

 **Answer:** 30%

This information is not in clear, non-software language, and does not pertain to users' investments' safety.

Guidance:

- | | |
|------|--|
| 100% | All the contracts are immutable |
| 90% | Description relates to investments safety and updates in clear, complete non-software 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 control is well documented, though there is no mention of testing. [The GitHub repository](#) seems to be mostly deprecated.

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	Maker 0.7		
		Points	Answer	
	Total	260	211.3	81%
Code and Tests				

Code and team		91	
1) Are the executing code addresses readily available? (%)	20	20%	4
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 requirements?	15	100%	15
9) Are there sufficiently detailed comments for all functions with logic?	5	36%	1.8
10) Is it possible to trace from software documentation to the source code?	10	60%	6
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	100%	20
12) Code coverage (Covers all the deployed lines of code, or executable statements) (%)	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	100%	5
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 administrator account? (%)	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 access?	10	80%	8
Section Scoring			
Code and Team	50	68%	
Documentation	45	84%	
Testing	50	75%	
Security	80	96%	
Access Controls	35	71%	

Executing Code Appendix

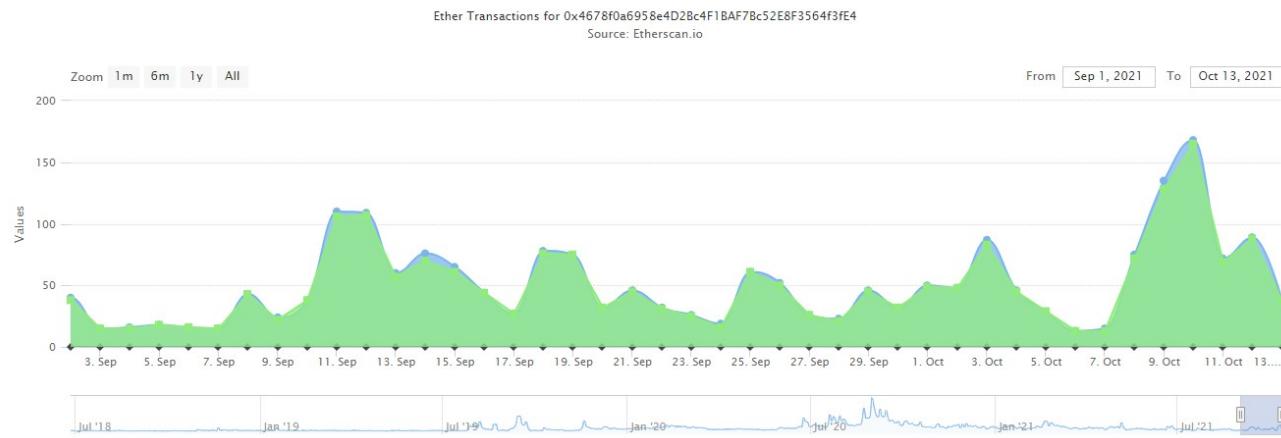
```
1 // SPDX-License-Identifier: GPL-3.0-or-later
2 pragma solidity 0.6.12;
3
4 contract Addresses {
5
6     mapping (bytes32 => address) public addr;
7
8     constructor() public {
9         addr["CHANGELOG"] = 0xdA0Ab1e0017DEbCd72Be8599041a2aa3bA7e740F;
10        addr["MULTICALL"] = 0x5e227AD1969Ea493B43F840cff78d08a6fc17796;
11        addr["FAUCET"] = 0x000000000000000000000000000000000000000000000000000000000000000;
12        addr["MCD_DEPLOY"] = 0xbbaa65281c2FA2baAcB2cb550BA051525A480D3F4;
13        addr["FLIP_FAB"] = 0x4ACdbde9dd0d00b36eC2050E805012b8Fc9974f2b;
14        addr["CLIP_FAB"] = 0x0716F25fBaAae9b63803917b6125c10c313dF663;
15        addr["CALC_FAB"] = 0xE1820A2780193d74939CcA104087CADd6c1aA13A;
16        addr["LERP_FAB"] = 0x00B416da876fe42dd02813da435Cc030F0d72434;
17        addr["MCD_GOV"] = 0x9f8F72aA9304c8B593d555F12eF6589cC3A579A2;
18        addr["ISOL_GUARD"] = 0x1c5FB2688267A018536B78520B80415270236D8E5;
```

```

10     addr[ "GOV_GUARD" ]           = 0x00ECD0002C/H910T00D/0E2U0000L7Z/Y2C000R0o;
19     addr[ "MCD_ADM" ]             = 0x0a3f6849f78076aefaf113F5BED87720274dDC0;
20     addr[ "VOTE_DELEGATE_PROXY_FACTORY" ] = 0xD897F108670903D1d6070fcf818f9db3615AF272;
21     addr[ "MCD_VAT" ]              = 0x35D1b3F3D7966A1DFe207aa4514C12a259A0492B;
22     addr[ "MCD_JUG" ]              = 0x19c0976f590D67707E62397C87829d896Dc0f1F1;
23     addr[ "MCD_CAT" ]              = 0xa5679C04fc3d9d8b0AaB1F0ab83555b301cA70Ea;
24     addr[ "MCD_DOG" ]              = 0x135954d155898D42C90D2a57824C690e0c7BEf1B;
25     addr[ "MCD_VOL" ]              = 0xA950524441892A31ebddF91d3cEEFa04BF454466;
26     addr[ "MCD_JOIN_DAI" ]         = 0x9759A6Ac90977b93B58547b4A71c78317f391A28;
27     addr[ "MCD_FLAP" ]             = 0xC4269cC7acDEdC3794b221aA4D9205F564e27f0d;
28     addr[ "MCD_FLOP" ]             = 0xA41B6EF151E06da0e34B009B86E828308986736D;
29     addr[ "MCD_PAUSE" ]            = 0xE286431454714F511008713973d3B053A2d38f3;
30     addr[ "MCD_PAUSE_PROXY" ]       = 0xBE8E3e3618f7474F8cB1d074A26afFef007E98FB;
31     addr[ "MCD_GOV_ACTIONS" ]       = 0x4F5f0933158569c026d617337614d00Ee6589B6E;
32     addr[ "MCD_DAI" ]              = 0x68175474E89094C44Da98b954EedeAC495271d0F;
33     addr[ "MCD_SPOT" ]              = 0x65C79FcB50Ca15948025960e539eD7A9a6D434A3;
34     addr[ "MCD_POT" ]              = 0x197E90f9FAD81970bA7976f33CbD77088E5D7cf7;
35     addr[ "MCD_END" ]              = 0xBB856d1742FD182a90239D7AE85706C2FE4e5922;
36     addr[ "MCD_ESM" ]              = 0x29CfBd381043D00a98fD9904a431015Fef07af2f;
37     addr[ "PROXY_ACTIONS" ]         = 0x82ecD135Dce65Fbc6DbD0e4237E0AF93FFD5038;

```

Code Used Appendix



Example Code Appendix

```

1 pragma solidity >=0.6.12;
2
3 interface VatLike {
4     function move(address,address,uint256) external;
5     function flux(bytes32,address,address,uint256) external;
6     function ilks(bytes32) external returns (uint256, uint256, uint256, uint256, uint256);
7     function suck(address,address,uint256) external;
8 }
9
10 interface PipLike {
11     function peek() external returns (bytes32, bool);
12 }
13
14 interface SpotterLike {

```

```

15     function par() external returns (uint256);
16     function ilks(bytes32) external returns (PipLike, uint256);
17 }
18
19 interface DogLike {
20     function chop(bytes32) external returns (uint256);
21     function digs(bytes32, uint256) external;
22 }
23
24 interface ClipperCallee {
25     function clipperCall(address, uint256, uint256, bytes calldata) external;
26 }
27
28 interface AbacusLike {
29     function price(uint256, uint256) external view returns (uint256);
30 }
31
32 contract Clipper {
33     // --- Auth ---
34     mapping (address => uint256) public wards;
35     function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
36     function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
37     modifier auth {
38         require(wards[msg.sender] == 1, "Clipper/not-authorized");
39         _;
40     }
41
42     // --- Data ---
43     bytes32 immutable public ilk;    // Collateral type of this Clipper
44     VatLike immutable public vat;   // Core CDP Engine
45
46     DogLike      public dog;       // Liquidation module
47     address      public vow;      // Recipient of dai raised in auctions
48     SpotterLike  public spotter;  // Collateral price module
49     AbacusLike   public calc;    // Current price calculator
50
51     uint256 public buf;          // Multiplicative factor to increase starting price
52     uint256 public tail;         // Time elapsed before auction reset
53     uint256 public cusp;         // Percentage drop before auction reset
54     uint64  public chip;         // Percentage of tab to suck from vow to incentivize keepers
55     uint192 public tip;          // Flat fee to suck from vow to incentivize keepers
56     uint256 public chost;        // Cache the ilk dust times the ilk chop to prevent excessive SI
57
58     uint256  public kicks;        // Total auctions
59     uint256[] public active;     // Array of active auction ids
60
61     struct Sale {
62         uint256 pos;   // Index in active array
63         uint256 tab;   // Dai to raise      [rad]
64         uint256 lot;   // collateral to sell [wad]
65         address usr;  // Liquidated CDP
66         uint96 tic;   // Auction start time
67         uint256 top;   // Starting price      [ray]

```

```
68     }
69     mapping(uint256 => Sale) public sales;
70
71     uint256 internal locked;
72
73     // Levels for circuit breaker
74     // 0: no breaker
75     // 1: no new kick()
76     // 2: no new kick() or redo()
77     // 3: no new kick(), redo(), or take()
78     uint256 public stopped = 0;
79
80     // --- Events ---
81     event Rely(address indexed usr);
82     event Deny(address indexed usr);
83
84     event File(bytes32 indexed what, uint256 data);
85     event File(bytes32 indexed what, address data);
86
87     event Kick(
88         uint256 indexed id,
89         uint256 top,
90         uint256 tab,
91         uint256 lot,
92         address indexed usr,
93         address indexed kpr,
94         uint256 coin
95     );
96     event Take(
97         uint256 indexed id,
98         uint256 max,
99         uint256 price,
100        uint256 owe,
101        uint256 tab,
102        uint256 lot,
103        address indexed usr
104    );
105    event Redo(
106        uint256 indexed id,
107        uint256 top,
108        uint256 tab,
109        uint256 lot,
110        address indexed usr,
111        address indexed kpr,
112        uint256 coin
113    );
114
115    event Yank(uint256 id);
116
117    // --- Init ---
118    constructor(address vat_, address spotter_, address dog_, bytes32 ilk_) public {
119        vat      = VatLike(vat_);
120        spotter = SpotterLike(spotter_);
```

```

121     dog      = DogLike(dog_);
122     ilk      = ilk_;
123     buf      = RAY;
124     wards[msg.sender] = 1;
125     emit Rely(msg.sender);
126 }
127
128 // --- Synchronization ---
129 modifier lock {
130     require(locked == 0, "Clipper/system-locked");
131     locked = 1;
132     _;
133     locked = 0;
134 }
135
136 modifier isStopped(uint256 level) {
137     require(stopped < level, "Clipper/stopped-incorrect");
138     _;
139 }
140
141 // --- Administration ---
142 function file(bytes32 what, uint256 data) external auth lock {
143     if (what == "buf")           buf = data;
144     else if (what == "tail")     tail = data;           // Time elapsed before auction
145     else if (what == "cusp")     cusp = data;          // Percentage drop before auction
146     else if (what == "chip")    chip = uint64(data);   // Percentage of tab to incentive
147     else if (what == "tip")     tip = uint192(data);  // Flat fee to incentivize keepers
148     else if (what == "stopped") stopped = data;        // Set breaker (0, 1, 2, or 3)
149     else revert("Clipper/file-unrecognized-param");
150     emit File(what, data);
151 }
152 function file(bytes32 what, address data) external auth lock {
153     if (what == "spotter") spotter = SpotterLike(data);
154     else if (what == "dog")  dog = DogLike(data);
155     else if (what == "vow")  vow = data;
156     else if (what == "calc") calc = AbacusLike(data);
157     else revert("Clipper/file-unrecognized-param");
158     emit File(what, data);
159 }
160
161 // --- Math ---
162 uint256 constant BLN = 10 ** 9;
163 uint256 constant WAD = 10 ** 18;
164 uint256 constant RAY = 10 ** 27;
165
166 function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
167     z = x <= y ? x : y;
168 }
169 function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
170     require((z = x + y) >= x);
171 }
172 function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
173     require((z = x - y) <= x);

```

```

173     require(y >= 0, "Clipper/yy<0");
174 }
175 function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
176     require(y == 0 || (z = x * y) / y == x);
177 }
178 function wmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
179     z = mul(x, y) / WAD;
180 }
181 function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
182     z = mul(x, y) / RAY;
183 }
184 function rdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
185     z = mul(x, RAY) / y;
186 }
187
188 // --- Auction ---
189
190 // get the price directly from the OSM
191 // Could get this from rmul(Vat.ilks(ilk).spot, Spotter.mat()) instead, but
192 // if mat has changed since the last poke, the resulting value will be
193 // incorrect.
194 function getFeedPrice() internal returns (uint256 feedPrice) {
195     (PipLike pip, ) = spotter.ilks(ilk);
196     (bytes32 val, bool has) = pip.peek();
197     require(has, "Clipper/invalid-price");
198     feedPrice = rdiv(mul(uint256(val), BLN), spotter.par());
199 }
200
201 // start an auction
202 // note: trusts the caller to transfer collateral to the contract
203 // The starting price `top` is obtained as follows:
204 //
205 //     top = val * buf / par
206 //
207 // Where `val` is the collateral's unitary value in USD, `buf` is a
208 // multiplicative factor to increase the starting price, and `par` is a
209 // reference per DAI.
210 function kick(
211     uint256 tab, // Debt [rad]
212     uint256 lot, // Collateral [wad]
213     address usr, // Address that will receive any leftover collateral
214     address kpr // Address that will receive incentives
215 ) external auth lock isStopped(1) returns (uint256 id) {
216     // Input validation
217     require(tab > 0, "Clipper/zero-tab");
218     require(lot > 0, "Clipper/zero-lot");
219     require(usr != address(0), "Clipper/zero-usr");
220     id = ++kicks;
221     require(id > 0, "Clipper/overflow");
222
223     active.push(id);
224
225     sales[id].pos = active.length - 1;

```

```

226
227     sales[id].tab = tab;
228     sales[id].lot = lot;
229     sales[id].usr = usr;
230     sales[id].tic = uint96(block.timestamp);
231
232     uint256 top;
233     top = rmul(getFeedPrice(), buf);
234     require(top > 0, "Clipper/zero-top-price");
235     sales[id].top = top;
236
237     // incentive to kick auction
238     uint256 _tip = tip;
239     uint256 _chip = chip;
240     uint256 coin;
241     if (_tip > 0 || _chip > 0) {
242         coin = add(_tip, wmul(tab, _chip));
243         vat.suck(vow, kpr, coin);
244     }
245

```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	16	3293	418	756	2119	457

Comments to Code $756/2119 = 36\%$

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	14	6167	1001	456	4710	41

Tests to Code $4710/2119 = 222\%$