Cage Keeper

A Keeper to facilitate Emergency Shutdown

Introduction

Thecage-keeper is used to help facilitateEmergency Shutdown of theMaker Protocol . Emergency shutdown is an involved, deterministic process, requiring interaction from all user types: Vault owners, Dai holders, Redemption keepers, MKR governors, and other Maker Protocol Stakeholders. A high level overview is as follows:

1. System Caged
2.
    - The Emergency Security Module(ESM)
3. callsEnd.cage()
4. function, which freezes the USD price for each collateral type as well as many parts of the system.
5. Processing Period
6.
    - Next, Vault owners interact with End to settle their Vault and withdraw excess collateral. Auctions are left to conclude or are yanked before Dai redemption.
7. Dai Redemption
8.
    - After the processing period durationEnd.wait
9. has elapsed, Vault settlement and all Dai generating processes (auctions) are assumed to have concluded. At this point, Dai holders can begin to claim a proportional amount of each collateral type at a fixed rate.
10.

To prevent a race-condition for Dai holders during Step 3, it's imperative that any Vaults having a collateralization ratio of less than 100% at Step 1 must be processed during Step 2. The owner of an underwater Vault would not receive excess collateral, so they lack an incentive toskim their position in theEnd contract. Thus, it is the responsibility of a MakerDAO Stakeholder (MKR holders, large Dai holders, etc) to ensure the system facilitates a Dai redemption phase without a time variable. Thecage-keeper is a tool to help stakeholders carry out this responsibility.

Prerequisites

The following section assumes familiarity with Emergency Shutdown. Good places to start is the Emergency Shutdown Module in Section 3 and Section 4 of theMaker Protocol 101 as well as a more thorough,technical description . Functions mentioned are from the implementation contained by theEnd contract, which islocated here .

To be consistent with the Protocol's technical terminology for the rest of this description:

- urn
- = Vault
- ilk
- = Collateral Type
-

Architecture

?

Thecage-keeper directly interacts with theEnd ,Flopper andFlapper contracts.

The central goal of thecage-keeper is to process all under-collateralizedurns . This accounting step is performed withinEnd.skim() , and since it is surrounded by other required/important steps in the Emergency Shutdown, a first iteration of this keeper will help to call most of the other public function calls within theEnd contract.

As can be seen in the above flowchart, the keeper checks if the system has been caged before attempting toskim all underwater urns andskip all flip auctions. After the processing period has been facilitated and theEnd.wait wait time has been reached, it will transition the system into the Dai redemption phase of Emergency Shutdown by callingEnd.thaw() andEnd.flow() . This first iteration of this keeper is naive, as it assumes it's the only keeper and attempts to account for all urns, ilks, and auctions. Because of this, it's important that the keeper's address has enough ETH to cover the gas costs involved with sending numerous transactions. Any transaction that attempts to call a function that's already been invoked by another Keeper/user would simply fail.

Operation

This keeper can either run continuously on a local/virtual machine or be run when the operator becomes aware of Emergency Shutdown. A sample startup script is shown below. The keeper's Ethereum address should have enough ETH to cover gas costs and is a function of the protocol's state at the time of shutdown (i.e. more urns toskim means more required

ETH to cover gas costs). When new collateral types are added to the protocol, the operator should pull the latest version of the keeper, which would include contracts associated with the aforementioned collateral types.

After thecage-keeper facilitates the processing period, it can be turned off untilEnd.wait is nearly reached. Then, at that point, the operator would pass in the--previous-cage argument during keeper start in order to bypass the feature that supports the processing period.

Installation

This project usesPython 3.6.2 .

In order to clone the project and install required third-party packages please execute:

```

Copy git clone https://github.com/makerdao/cage-keeper.git cd cage-keeper git submodule update --init --recursive ./install.sh

```

For some known Ubuntu and macOS issues see thepymaker README.

Sample Startup Script

Make a run-cage-keeper.sh to easily spin up the cage-keeper.

```

Copy

# !/bin/bash

/full/path/to/cage-keeper/bin/cage-keeper \ --rpc-host 'sample.ParityNode.com' \ --network 'kovan' \ --eth-from '0xABCAddress' \ --eth-key 'key_file=/full/path/to/keystoreFile.json,pass_file=/full/path/to/passphrase/file.txt' \ --vat-deployment-block 14374534

```

Testing

Prerequisites:

- Downloaddocker and docker-compose
-

This project usespytest for unit testing. Testing of Multi-collateral Dai is performed on a Dockerized local testchain included intests\config .

In order to be able to run tests, please install development dependencies first by executing:

```

Copy pip3 install -r requirements-dev.txt

```

You can then run all tests with :

```

Copy ./test.sh

```

License

SeeCOPYING file

Support

If you have questions regarding Cage Keepers, please reach out to us on the#keeper channel onchat.makerdao.com .