Auction Keeper Bot Setup Guide

Walkthrough how to set up your own Auction Keeper Level: Intermediate

Estimated Time: 60 minutes

Audience: Developers

Overview

The Maker Protocol, which powers Multi Collateral Dai (MCD), is a smart contract based system that backs and stabilizes the value of Dai through a dynamic combination of Vaults (formerly known as CDPs), autonomous feedback mechanisms, and incentivized external actors. To keep the system in a stable financial state, it is important to prevent both debt and surplus from building up beyond certain limits. This is where Auctions and Auction Keepers come in. The system has been designed so that there are three types of Auctions in the system: Surplus Auctions, Debt Auctions, and Collateral Auctions. Each auction is triggered as a result of specific circumstances.

Auction Keepers are external actors that are incentivized by profit opportunities to contribute to decentralized systems. In the context of the Maker Protocol, these external agents are incentivized to automate certain operations around the Ethereum blockchain. This includes:

- Seeking out opportunities and starting new auctions
- Detect auctions started by other participants
- Bid on auctions by converting token prices into bids
-

More specifically, Keepers participate as bidders in the Debt and Collateral Auctions when Vaults are liquidated and auction-keeper enables the automatic interaction with these MCD auctions. This process is automated by specifying bidding models that define the decision making process, such as what situations to bid in, how often to bid, how high to bid etc. Note that bidding models are created based on individually determined strategies.

Learning Objectives

This guide's purpose is to provide a walkthrough of how to useauction-keeper and interact with a Kovan deployment of the Multi Collateral Dai (MCD) smart contracts. More specifically, the guide will showcase how to set up and run an Auction Keeper bot for yourself. After going through this guide, you will achieve the following:

- Learn about Auction Keepers and how they interact with the Maker Protocol
- Understand bidding models
- Get your own auction keeper bot running on the Kovan testnet
-

Guide Agenda

This guide will show how to use the auction-keeper to interact with the Kovan deployment of the MCD smart contracts. More specifically, the guide will showcase how to go through the following stages of setting up and running an Auction Keeper bot:

1. Introduction
2. Bidding Models
3.
    - Starting and stopping bidding models
4.
    - Communicating with bidding models
5. *
6. Setting up the Keeper Bot (Flip Auction Keeper)
7.
    - Prerequisites
8.
    - Installation
9. *
10. Running your Keeper Bot (Usage)
11.
    - Keeper Limitations
12. *
13. Accounting
14.
    - Getting MCD K-DAI
15.
    - Getting MCD K-MKR
16.
    - Getting MCD Collateral Tokens
17. *
18. Testing
19. Support
20.

We are proud to say that since the Maker Protocol is an open-source platform, all of the code we have created to run the Keeper bot is free and accessible to all.

1. Introduction

Auction Keepers participate in auctions as a result of liquidation events and thereby acquire collateral at attractive prices. Anauction-keeper can participate in three different types of auctions:

1. Collateral Auction (flip)
2. Surplus Auction (flap)
3. Debt Auction (flop)
4.

Auction Keepers have the unique ability to plug in externalbidding models , which communicate information to the Keeper on when and how high to bid (these types of Keepers can be left safely running in the background). Shortly after an Auction Keeper notices or starts a new auction, it will spawn a new instance of abidding model and act according to its specified instructions. Bidding models will be automatically terminated by the Auction Keeper the moment the auction expires.

Note:

Auction Keepers will automatically calldeal (claiming a winning bid / settling a completed auction) if the Keeper's address won the auction.

Auction Keeper Architecture

As mentioned above, Auction Keepers directly interact withFlipper ,Flapper andFlopper auction contracts deployed to the Ethereum mainnet. All decisions which involve pricing details are delegated to thebidding models . The Bidding models are simply executable strategies, external to the mainauction-keeper process. This means that the bidding models themselves do not have to know anything about the Ethereum blockchain and its smart contracts, as they can be implemented in basically any programming language. However, they do need to have the ability to read and write JSON documents, as this is how they communicate/exchange withauction-keeper . It's important to note that as a developer running an Auction Keeper, it is required that you have basic knowledge on how to properly start and configure the auction-keeper. For example, providing startup parameters as keystore / password are required to setup and run a Keeper. Additionally, you should be familiar with the MCD system, as the model will receive auction details from auction-keeper in the form of a JSON message containing keys such as lot, beg, guy, etc.

Simple Bidding Model Example:

A simple bidding model could be a shell script which echoes a fixed price (further details below).

The Purpose of Auction Keepers

The main purpose of Auction Keepers are:

- To discover new opportunities and start new auctions.
- To constantly monitor all ongoing auctions.
- To detect auctions started by other participants.
- To Bid on auctions by converting token prices into bids.
- To ensure that instances ofbidding model
- are running for each auction type as well as making sure the instances match the current status of their auctions. This ensure that Keepers are bidding according to decisions outlined by the bidding model.
-

The auction discovery and monitoring mechanisms work by operating as a loop, which initiates on every new block and enumerates all auctions from1 tokicks . When this occurs, even when thebidding model decides to send a bid, it will not be processed by the Keeper until the next iteration of that loop. It's important to note that theauction-keeper not only monitors existing auctions and discovers new ones, but it also identifies and takes opportunities to create new auctions.

1. Bidding Models

Starting and Stopping Bidding Models

Auction Keeper maintains a collection of child processes, as eachbidding model is its own dedicated process. New processes (newbidding model instances) are spawned by executing a command according to the--model command-line parameter. These processes are automatically terminated (viaSIGKILL) by the keeper shortly after their associated auction expires. Whenever thebidding model process dies, it gets automatically re-spawned by the Keeper.

Example:

```

Copy bin/auction-keeper --model '../my-bidding-model.sh' [...]

```

Communicating with bidding models

Auction Keepers communicate withbidding models via their standard input/standard output. Once the process has started and every time the auction state changes, the Keeper sends a one-line JSON document to thestandard input of thebidding model.

A sample JSON message sent from the keeper to the model looks like the:

```

Copy {"id": "6", "flapper": " 0xf0afc3108bb8f196cf8d076c8c4877a4c53d4e7c ", "bid": "7.142857142857142857", "lot": "10000.000000000000000000", "beg": "1.050000000000000000", "guy": " 0x00531a10c4fbd906313768d277585292aa7c923a ", "era": 1530530620, "tic": 1530541420, "end": 1531135256, "price": "1400.000000000000000028"}

```

Glossary (Bidding Models):

- id
-
    - auction identifier.

- flipper
-
    - Ethereum address of theFlipper
- contract (only forflip
- auctions).
- flapper
-
    - Ethereum address of theFlapper
- contract (only forflap
- auctions).
- flopper
-
    - Ethereum address of theFlopper
- contract (only forflop
- auctions).
- bid
-
    - current highest bid (will go up forflip
- andflap
- auctions).
- lot
-
    - amount being currently auctioned (will go down forflip
- andflop
- auctions).
- tab
-
    - bid value (not to be confused with the bid price) which will cause the auction to enter thedent
- phase (only forflip
- auctions).
- beg
-
    - minimum price increment (1.05
- means minimum 5% price increment).
- guy
-
    - Ethereum address of the current highest bidder.
- era
-
    - current time (in seconds since the UNIX epoch).
- tic
-
    - time when the current bid will expire (None
- if no bids yet).
- end
-
    - time when the entire auction will expire (end is set to0
- is the auction is no longer live).
- price
-
    - current price being tendered (can beNone
- if price is infinity).
-

Bidding models should never make an assumption that messages will be sent only when auction state changes. It is perfectly fine for theauction-keeper to periodically send the same message(s) tobidding models .

At the same time, theauction-keeper reads one-line messages from thestandard output of thebidding model process and tries to parse them as JSON documents. It will then extract the two following fields from that document:

- price
-
    - the maximum (forflip
- andflop
- auctions) or the minimum (forflap
- auctions) price the model is willing to bid.
- gasPrice
- (optional) - gas price in Wei to use when sending a bid.
-

An example of a message sent from the Bidding Model to the Auction Keeper may look like:

```

Copy {"price": "150.0", "gasPrice": 7000000000}

```

In the case of when Auction Keepers and Bidding Models communicate in terms of prices, it is the MKR/DAI price (forflap andflop auctions) or the collateral price expressed in DAI forflip auctions (for example, OMG/DAI).

Any messages written by a Bidding Model tostderr (standard error) will be passed through by the Auction Keeper to its logs. This is the most

convenient way of implementing logging from Bidding Models.

1. Setting up the Auction Keeper Bot (Installation)

Prerequisite

- Git
- Python v3.6.6
- virtualenv
-
    - This project requiresvirtualenv
-
    - to be installed if you want to use Maker's python tools. This helps to ensure that you are running the right version of python as well as check that all of the pip packages that are installed in theinstall.sh
-
    - are in the right place and have the correct versions.
- *
- X-code
- (for Macs)
- Docker-Compose
-

Getting Started

Installation from source:

1. Clone the
2. auction-keeper
3. repository:
4.

```

Copy git clone https://github.com/makerdao/auction-keeper.git

```

1. Switch into the
2. auction-keeper
3. directory:
4.

```

Copy cd auction-keeper

```

1. Install required third-party packages:
2.

```

Copy git submodule update --init --recursive

```

1. Set up the virtual env and activate it:
2.

```

Copy python3 -m venv _virtualenv source _virtualenv/bin/activate

```

1. Install requirements:

```

Copy pip3 install -r requirements.txt

```

Potential Errors:

- Needing to upgrade pip version to 19.2.2:
-
    - Fix by runningpip install --upgrade pip
-
    - .
- *
-

For other known Ubuntu and macOS issues please visit thepymaker README.

1. Running your Keeper Bot

The Kovan version runs on theKovan Release 1.0.2

To change to your chosen version of the kovan release, copy/paste your preferred contract addresses inkovan-addresses.json inlib/pymaker/config/kovan-addresses.json

1. Creating your bidding model (an example detailing the simplest possible bidding model)

The stdout (standard output) provides a price for the collateral (forflip auctions) or MKR (forflap andflop auctions). Thesleep locks the price in place for a minute, after which the keeper will restart the price model and read a new price (consider this your price update interval).

The simplest possiblebidding model you can set up is when you use a fixed price for each auction. For example:

```

Copy

# !/usr/bin/env bash

echo "{\"price\": \"150.0\"}" # put your desired fixed price amount here sleep 60 # locking the price for a 60 seconds period

```

Once you have created your bidding model, save it asmodel-eth.sh (or whatever name you feel seems appropriate).

1. Setting up an Auction Keeper for a Collateral (Flip) Auction

Collateral Auctions will be the most common type of auction that the community will want to create and operate Auction keepers for. This is due to the fact that Collateral auctions will occur much more frequently than Flap and Flop auctions.

Example (Flip Auction Keeper):

- This example/process assumes that the user has an already existing shell script that manages their environment and connects to the Ethereum blockchain and that you have some Dai and Kovan ETH in your wallet. If you don't have any balance, check the section below on how to get some.
-

An example on how to set up your environment: asmy_environment.sh

```

Copy SERVER_ETH_RPC_HOST=https://your-ethereum-node SERVER_ETH_RPC_PORT=8545 ACCOUNT_ADDRESS=0x16Fb96a5f-your-eth-address-70231c8154saf
ACCOUNT_KEY="key_file=/Users/username/Documents/Keeper/accounts/keystore,pass_file=/Users/username/Documents/keeper/accounts/pass"

```

SERVER_ETH_RPC_HOST - Should not be an infura node, as it doesn't provide all the functionality that the python script needsACCOUNT_KEY - Should have the absolute path to the keystore and password file. Define the path as shown above, as the python script will parse through both the keystore and password files.

```

Copy

# !/bin/bash

dir="(dirname "0")"

source my_environment.sh # Set the RPC host, account address, and keys. source _virtualenv/bin/activate # Run virtual environment

# Allows keepers to bid different prices

MODEL=1

bin/auction-keeper \ --rpc-host {SERVER_ETH_RPC_HOST:?} \ --rpc-port {SERVER_ETH_RPC_PORT?:} \ --rpc-timeout 30 \ --eth-from {ACCOUNT_ADDRESS?:} \ --eth-key {ACCOUNT_KEY?:} \ --type flip \ --ilk ETH-A \ --from-block 14764534 \ --vat-dai-target 1000 \ --model {dir} {MODEL} \ 2> >(tee -a -i auction-keeper-flip-ETH-A.log >&2)

```

Once finalized, you should save your script to run your Auction Keeper asflip-eth-a.sh (or something similar to identify that this Auction Keeper is for a Flip Auction). In addition, make sure to verify the above copy+pasted script doesn't create extra spaces or characters on pasting+saving in your editor. You will notice an error when running it later below otherwise.

Important Note about Running Auction Keepers on the Ethereum Mainnet!

- If you get to the point where the auction keeper bot is not accepting mainnet as a valid argument, this is because there is nonetwork
- parameter. To fix this, just omit that parameter.
-

Other Notes:

- All Collateral types (ilk
- 's) combine the name of the token and a letter corresponding to a set of risk parameters. For example, as you can see above, the example uses ETH-A. Note that ETH-A and ETH-B are two different collateral types for the same underlying token (WETH) but have different risk parameters.
- For the MCD addresses, we simply pass--network mainnet|kovan
- in and it will load the required JSON files bundled within auction-keeper (or pymaker).
-

1. Passing the bidding the model as an argument to the Keeper script

2. Confirm that both your bidding model (model-eth.sh) and your script (flip-eth-a.sh) to run your Auction Keeper are saved.

3. The next step is tochmod +x
4. both of them.
5. Lastly, runflip-eth-a.sh model-eth.sh
6. to pass your bidding model into your Auction Keeper script.
7.

Example of a working keeper: After running the./flip-eth-a.sh model-eth.sh command you will see an output like this:

```

Copy 019-10-31 13:33:08,703 INFO Keeper connected to RPC connection https://parity0.kovan.makerfoundation.com:8545 2019-10-31 13:33:08,703 INFO Keeper operating as 0x16Fb96a5fa0427Af0C8F7cF1eB4870231c8154B6 2019-10-31 13:33:09,044 INFO Executing keeper startup logic 2019-10-31 13:33:09,923 INFO Sent transaction DSToken('0x1D7e3a1A65a367db1D1D3F51A54aC01a2c4C92ff').approve(address,uint256) ('0x9E0d5a6a836a6C323Cf45Eb07Cb40CFc81664eec', 115792089237316195423570985008687907853269984665640564039457584007913129639935) with nonce=1257, gas=125158, gas_price=default (tx_hash=0xc935e3a95e5d0839e703dd69b6cb2d8f9a9d3d5cd34571259e36e771ce2201b7) 2019-10-31 13:33:12,964 INFO Transaction DSToken('0x1D7e3a1A65a367db1D1D3F51A54aC01a2c4C92ff').approve(address,uint256) ('0x9E0d5a6a836a6C323Cf45Eb07Cb40CFc81664eec', 115792089237316195423570985008687907853269984665640564039457584007913129639935) was successful (tx_hash=0xc935e3a95e5d0839e703dd69b6cb2d8f9a9d3d5cd34571259e36e771ce2201b7) 2019-10-31 13:33:13,152 WARNING Insufficient balance to maintain Dai target; joining 91.319080635247876480 Dai to the Vat 2019-10-31 13:33:13,751 INFO Sent transaction .join('0x16Fb96a5fa0427Af0C8F7cF1eB4870231c8154B6', 91319080635247876480) with nonce=1258, gas=165404, gas_price=default (tx_hash=0xcce12af8d27f9d6185db4b359b8f3216ee783250a1f3b3921256efabb63e22b0) 2019-10-31 13:33:16,491 INFO Transaction .join('0x16Fb96a5fa0427Af0C8F7cF1eB4870231c8154B6', 91319080635247876480) was successful (tx_hash=0xcce12af8d27f9d6185db4b359b8f3216ee783250a1f3b3921256efabb63e22b0) 2019-10-31 13:33:16,585 INFO Dai token balance: 0.000000000000000000, Vat balance: 91.319080635247876480133691494546726938904901298 2019-10-31 13:33:16,586 INFO Watching for new blocks 2019-10-31 13:33:16,587 INFO Started 1 timer(s)

```

Now the keeper is actively listening for any action. If it sees an undercollateralized position, then it will try to bid for it.

Auction Keeper Arguments Explained

To participate in all auctions, a separate keeper must be configured forflip of each collateral type, as well as one forflap and another forflop .

1. --type
2.
    - the type of auction the keeper is used for. In this particular scenario, it will be set toflip
3. .
4. --ilk
5.
    - the type of collateral.
6. --addresses
7.
    - .json of all of the addresses of the MCD contracts as well as the collateral types allowed/used in the system.
8. --vat-dai-target
9.
    - the amount of DAI which the keeper will attempt to maintain in the Vat, to use for bidding. It will rebalance it upon keeper startup and upondeal
10. ing an auction.
11. --model
12.
    - the bidding model that will be used for bidding.
13. --from-block
14. to the block where the first urn was created to instruct the keeper to use logs published by the vat contract to bulid a list of urns, and then check the status of each urn.
15.

Callbin/auction-keeper --help for a complete list of arguments.

Auction Keeper Limitations

- If an auction starts before the auction Keeper has started, the Keeper will not participate in the auction until the next block has been mined.
- Keepers do not explicitly handle global settlement (End
-). If global settlement occurs while a winning bid is outstanding, the Keeper will not request ayank
- to refund the bid. The workaround is to callyank
- directly usingseth
- .
- There are some Keeper functions that incur gas fees regardless of whether a bid is submitted. This includes, but is not limited to, the following actions:
-
    - Submitting approvals.
-
    - Adjusting the balance of surplus to debt.
-
    - Biting a CDP or starting a flap or flop auction, even if insufficient funds exist to participate in the auction.
- *
- The Keeper will not check model prices until an auction officially exists. As such, it willkick
- ,flap
- , orflop
- in response to opportunities regardless of whether or not your DAI or MKR balance is sufficient to participate. This imposes a gas fee that must be paid.
-
    - After procuring more DAI, the Keeper must be restarted to add it to theVat
-
    - .
- *
-

1. Accounting

The Auction contracts exclusively interact with DAI (for all auctions types) and collateral (forflip auctions) in theVat . More explicitly speaking:

- The DAI that is used to bid on auctions is withdrawn from theVat
- .
- The Collateral and surplus DAI won at auction end is placed in theVat
- .
-

By default, all the DAI and collateral within youreth-from account isexit 'ed from the Vat and added to your account token balance when the Keeper is shut down. Note that this feature may be disabled using thekeep-dai-in-vat-on-exit andkeep-gem-in-vat-on-exit switches, respectively. The use of aneth-from account with an open CDP is discouraged, as debt will hinder the auction contracts' ability to access your DAI, and theauction-keeper 's ability toexit DAI from theVat .

When running multiple Auction Keepers using the same account, the balance of DAI in theVat will be shared across all of the Keepers. If using this feature, you should set--vat-dai-target to the same value for each Keeper, as well as sufficiently high in order to cover total desired exposure.

Note:

MKR used to bid onflap auctions is directly withdrawn from your token balance. The MKR won atflop auctions is directly deposited to your token balance.

Getting Kovan MCD DAI, MKR and other Collateral tokens

1. Getting MCD K-DAI (K-MCD 0.2.12 Release)

Contract address :0xb64964e9c0b658aa7b448cdbddfcdccab26cc584

1. Log into your MetaMask account from the browser extension. Add or confirm that the custom MCD K-DAI token is added to your list of tokens.
2.
    - This done by selecting "Add Token" and then by adding in the details under the "Custom token" option.
3. *
4. Head to Oasis Borrowhere
5. .
6.
    - Confirm that you are in fact on the Kovan Network before proceeding.
7. *
8. Connect your MetaMask account.
9. Approve the MetaMask connection.
10. Below the "Overview" button, find and select the plus sign button to start setting up your CDP.
11. Select the collateral type you want to proceed with and click "Continue".
12.
    - e.g. ETH-A
13. *
14. Deposit your K-ETH and generate K-DAI by selecting and inputing an amount of K-ETH and the amount of K-DAI you would like to generate. To proceed, click "Continue".
15.
    - e.g. Deposit 0.5 K-ETH and generate 100 DAI.
16. *
17. Click on the checkbox to confirm that you have read and accepted theTerms of Service
18. then click the "Create CDP" button.
19. Approve the transaction in your MetaMask extension.
20. Click the "Exit" button and wait for your CDP to be created.

21.

After all of these steps have been completed, you will have the generated MCD K-DAI and it will be present within your wallet. You can easily payback your DAI or generate more.

1. Getting MCD K-MKR (K-MCD 1.0.2 Release)

Contract address: 0xaaf64bfcc32d0f15873a02163e7e500671a4ffcd

This requires familiarity with Seth as well as having the tool set up on your local machine. If unfamiliar, use this guide to install and set it up.

Run the following command in Seth:

```

Copy seth send 0xcbd3e165ce589657fefd2d38ad6b6596a1f734f6 'gulp(address)' 0xaaf64bfcc32d0f15873a02163e7e500671a4ffcd

```

Address information:

- The0x94598157fcf0715c3bc9b4a35450cce82ac57b20
- address is the faucet that issues 1 MKR per request.
- The0xaaf64bfcc32d0f15873a02163e7e500671a4ffcd
- address is that of the MCD K-MKR token. It will issue 1 MKR.
-

Important Note: The faucet address and token addresses often change with each dss deployment. The current addresses displayed above are from the0.2.12 Release . Please visithttps://changelog.makerdao.com/ for the most updated release version.

Please refer to thisguide to obtain collateral test tokens for Kovan.

1. Getting MCD Collateral Tokens

1. Testing your Keeper

To help with the testing of your Auction Keeper, we have created a collection of python and shell scripts herein that may be used to testauction-keeper ,pymaker 's auction facilities, and relevant smart contracts indss . For more information about testing your Auction Keeper with your own testchain visittests/manual/README .

1. Support

We welcome any questions or concerns about the Auction Keepers in the#keeper channel in the Maker Chat.

Previous Auction Keepers Next Market Maker Keepers Last updated4 years ago On this page *Overview * Learning Objectives * Guide Agenda * 1. Introduction * Auction Keeper Architecture * The Purpose of Auction Keepers * 2. Bidding Models * Starting and Stopping Bidding Models * Communicating with bidding models * Glossary (Bidding Models): * 3. Setting up the Auction Keeper Bot (Installation) * Getting Started * 4. Running your Keeper Bot * Auction Keeper Limitations * 5. Accounting * Getting Kovan MCD DAI, MKR and other Collateral tokens * 6. Testing your Keeper * 7. Support

Export as PDF