Cat - Detailed Documentation

The Maker Protocol's Liquidation Agent * Contract Name: * cat.sol * Type/Category: * DSS *Associated MCD System Diagram * Contract Source * Etherscan *

1. Introduction (Summary)

TheCat is the system's liquidation agent, it enables keepers to mark positions as unsafe and sends them to auction.

?

1. Contract Details

Glossary (Cat)

Math

- mul(uint, uint)
- /rmul(uint, uint)
-
    - will revert on overflow or underflow
- bite(bytes32, address)
- will revert iflot
- orart
- are larger than or equal to 2^255.
- bite
- will not leave a Vault with debt and no collateral.
-

Auth

- wards
- are allowed to call protected functions (Administration andcage()
-)
-

Storage

- ilks
- storesIlk
- structs
-
    - Ilk
-
    - is the struct with the address of the collateral auction contract (flip
-
    -), the penalty for that collateral to be liquidated (chop
-
    -) and the maximum size of collateral that can be auctioned at once (dunk
-
    -).
- *
- live
- must be1
- for theCat
- tobite
- . (seecage
- in mechanisms)
- box
- the limit on the debt and penalty fees available for auction. [RAD]
- dunk
- ("debt chunk") amount of debt plus penalty fee per auction, in Dai. [RAD]
- vat
- address that conforms to aVatLike
- interface (seevat documentation
- for more info). It is set during the constructor andcannot be changed
- .
- vow

- address that conforms to aVowLike
- interface (seevow documentation
- for more info).
-

The values of all parameters here (exceptvat) are changeable by an address that isrely ed on. For instance, theEnd module should beauth ed to allow for it to callcage() and updatelive from 1 to 0. Governance (through anauth ed address) should be able to add collateral types toCat , update their parameters, and change thevow .

Unsafe

bite can be called at anytime but will only succeed if the Vault is unsafe. A Vault is unsafe when its locked collateral (ink) times its collateral's liquidation price (spot) is less than its debt (art times the fee for the collateralrate). Liquidation price is the oracle-reported price scaled by the collateral's liquidation ratio.

Events

- Bite
- : emitted when abite(bytes32, address)
- is successfully executed. Contains:
-
    - ilk
-
    - : Collateral
-
    - urn
-
    - : Vault address
-
    - ink
-
    - : seelot
-
    - inbite
-
    - art
-
    - : seeart
-
    - inbite
-
    - tab
-
    - : seetab
-
    - inbite
-
    - flip
-
    - : address of the auction contract
-
    - id
-
    - : ID of the auction in theFlipper
- *
-

1. Key Mechanisms & Concepts

cage()

- auth
- setslive
- to 0 (preventsbite
-). SeeEnd documentation
- for further description.
- Oncelive=0
- it cannot be set back to 1.
-

bite(bytes32 ilk, address urn)

- bytes32 ilk
- parameter represents the collateral type that is being bitten.
- address urn
- the address that identifies the Vault being bitten.
- returnsuint id
- which is the ID of the new auction in theFlipper
- .
- bite
- checks if the Vault is in an unsafe position and if it is, it starts a Flip auction for a piece of the collateral to cover a share of the debt.
-

The following is a line-by-line explanation of whatbite does.

```

Copy function bite(bytes32 ilk, address urn) public returns (uint id) { // Get the rate, spot, and dust from the ilk in the vat. (,uint256 rate,uint256 spot,,uint256 dust) = vat.ilks(ilk); // get the ink and art from the urn from the Vat. (uint256 ink, uint256 art) = vat.urns(ilk, urn);

// ensure End has not happened require(live == 1); // require the Vault to be unsafe (see definition above). require(spot > 0 && mul(ink, spot) < mul(art, rate), "Cat/not-unsafe");

// Loads the ilk data into memory as an optimization. Ilk memory milk = ilks[ilk]; // Declares a variable that will be assigned in the following scope. uint256 dart;

// Defines a new scope, this prevents a stack too deep error in solidity { // Calculate the available space in the box uint256 room = sub(box, litter);

// test whether the remaining space in the litterbox is dusty require(litter < box && room >= dust, "Cat/liquidation-limit-hit");

// Sets the amount of debt to be covered by the auction. // (smaller of either the amount of normalized debt, maximum debt chunk size, or space in the box) // divided by the rate, divided by the penalty fee. dart = min(art, mul(min(milk.dunk, room), WAD) / rate / milk.chop); }

// Takes the minimum of the collateral balance or the // amount of collateral represented by the debt to be covered uint256 dink = min(ink, mul(ink, dart) / art);

// Prevents no-collateral auctions require(dart > 0 && dink > 0 , "Cat/null-auction"); // Protects against int overflow when converting from uint to int require(dart <= 2**255 && dink <= 2**255, "Cat/overflow" );

// Called in this way, vat.grab will: // - Remove the dink and the dart from the bitten Vault (urn) // - Adds the collateral (dink) to the Cat's gem // - Adds the debt (dart) to the Vow's debt (vat.sin[vow]) // - Increases the total unbacked dai (vice) in the system // This may leave the CDP in a dusty state vat.grab( ilk, urn, address(this), address(vow), -int256(dink), -int256(dart) ); // Adds the debt to the debt-queue in Vow (Vow.Sin and Vow.sin[now]) vow.fess(mul(dart, rate));

{ // Avoid stack too deep // This calcuation will overflow if dart*rate exceeds ~10^14, // i.e. the maximum dunk is roughly 100 trillion DAI. // Multiplies the accumulated rate by the normalized debt to be covered // to get the total debt tab (debt + stability fee + liquidation penalty) for the auction. uint256 tab = mul(mul(dart, rate), milk.chop) / WAD; // Updates the amount of litter in the box litter = add(litter, tab);

// Calls kick on the collateral's Flipper contract. // tab is the total debt to be sent to auction // gal: address(vow) sets up the Vow as the recipient of the Dai income for this auction // bid: 0 indicates that this is the opening bid // This moves the collateral from the Cat's gem to the Flipper's gem in the Vat id = Kicker(milk.flip).kick({ urn: urn, gal: address(vow), tab: tab, lot: dink, bid: 0 }); }

// Emits an event about the bite to notify actors (for instance keepers) about the new auction emit Bite(ilk, urn, dink, dart, mul(dart, rate), milk.flip, id); }

```

Administration

Various file function signatures for administeringCat :

- Setting new vow (file(bytes32, address)
-)
- Setting new collateral (file(bytes32, bytes32, address)
-)

- Setting penalty or dunk size for collateral (file(bytes32, bytes32, uint)
-)
-

Usage

The primary usage will be for keepers to callbite on a Vault they believe to be unsafe in order to start the auction process.

1. Gotchas (Potential source of user error)

2. When theCat

3. is upgraded, there are multiple references to it that must be updated at the same time (End
4. ,Vat.rely
5. ,Vow.rely
6.). It must also rely on theEnd
7. , the system'spause.proxy()
8. AVat
9. upgrade will require a newCat
10. The Cat stores eachIlk
11. 's liquidation penalty and maximum auction size.
12. Each ilk will be initiated with thefile
13. for theirFlipper
14. ; however, auctions cannot start untilfile
15. is also called to set thechop
16. and thedunk
17. . Without these auctions for either 0 gems or 0 dai would be created by callingbite
18. on an unsafe Vault.
19. bite
20. needs to be calledn
21. times wheren = urn.ink / ilks[ilk].dunk
22. ifn > 1
23. . This allows for the possibility that the Vault becomes safe betweenbite
24. calls either through increased collateral (in value or quantity), or decreased debt.
25. Callingbite
26. returns the auctionid
27. and emits and event with theid
28. . This is required to bid in theFlipper
29. on the auction.
30.

1. Failure Modes (Bounds on Operating Conditions & External Risk Factors)

Coding Error

A bug in theCat could lead to loss (or locking) of Dai and Collateral by assigning it to an address that cannot recover it (i.e. a bad Vow address or an incorrectly programmed Flipper). The main coding failure mode ofCat is if it has a bug that causes auctions to cease to function. This would require upgrading the system to a correctedCat contract. If there is a bug inCat that reverts oncage then it would cause Shutdown could fail (until a correctCat is launched).

Feeds

TheCat relies indirectly on the price Feeds as it looks to theVat 's tracking of the collateral prices (spot) to determine Vault safety. If this system breaks down, it could lead to theft of collateral (too lowspot) or unbacked Dai (incorrectly highspot).

Governance

Governance can authorize and configure new collaterals forCat . This could lead to misconfiguration or inefficiencies in the system. Misconfiguration could causeCat not to operate properly or at all. For instance, if anIlk.dunk is set to be greater than 2**255 could allow for very, very large Vaults to be un-bite -able.

Inefficiencies in thedunk orchop could affect auctions. For instance, adunk that is too large or too small could lead to disincentives for keepers to participate in auctions. Achop that is too small would not sufficiently dis-incentivize risky Vaults and too large could lead to it being converted to bad debt. Further discussion of how the parameters could lead to system attacks is described in thisAuction Grinding paper .

Flipper

TheCat relies on an external Flipper contract to run the auction and moves the collateral from theCat to theFlipper contracts

in theVat . A faulty collateral auction contract could result in the loss of collateral or dai or non-functioning auctions.

Previous Liquidations 1.2 System (Deprecated)Next Flipper - Detailed Documentation Last updated3 years ago On this page *

Export as PDF