Vat - Detailed Documentation

The Maker Protocol's Core Accounting System * Contract Name: Vat.sol * Type/Category: * DSS —> Core System Accounting * [Associated MCD System Diagram](#) * [Contract Source](#) * [Etherscan](#) *

1. Introduction (Summary)

TheVat is the core Vault engine ofdss . It stores Vaults and tracks all the associated Dai and Collateral balances. It also defines the rules by which Vaults and balances can be manipulated. The rules defined in theVat are immutable, so in some sense, the rules in theVat can be viewed as the constitution ofdss .

?

1. Contract Details

Glossary (Vat - VaultEngine)

- gem
- : collateral tokens.
- dai
- : stablecoin tokens.
- sin
- : unbacked stablecoin (system debt, not belonging to anyurn
-).
- ilks
- : a mapping ofIlk
- types.
- Ilk
- : a collateral type.
-
    - Art
-
    - : total normalized stablecoin debt.
-
    - rate
-
    - : stablecoin debt multiplier (accumulated stability fees).
-
    - spot
-
    - : collateral price with safety margin, i.e. the maximum stablecoin allowed per unit of collateral.
-
    - line
-
    - : the debt ceiling for a specific collateral type.
-
    - dust
-
    - : the debt floor for a specific collateral type.
- *
- urns
- : a mapping ofUrn
- types.
- Urn
- : a specific Vault.
-
    - ink
-
    - : collateral balance.
-
    - art
-
    - : normalized outstanding stablecoin debt.
- *
- init
- : create a new collateral type.
- slip
- : modify a user's collateral balance.

- flux
- : transfer collateral between users.
- move
- : transfer stablecoin between users.
- grab
- : liquidate a Vault.
- heal
- : create / destroy equal quantities of stablecoin and system debt (vice
-).
- fold
- : modify the debt multiplier, creating / destroying corresponding debt.
- suck
- : mint unbacked stablecoin (accounted for withvice
-).
- Line
- : the total debt ceiling for all collateral types.
- frob
- : modify a Vault.
-
    - lock
-
    - : transfer collateral into a Vault.
-
    - free
-
    - : transfer collateral from a Vault.
-
    - draw
-
    - : increase Vault debt, creating Dai.
-
    - wipe
-
    - : decrease Vault debt, destroying Dai.
-
    - dink
-
    - : change in collateral.
-
    - dart
-
    - : change in debt.
- *
- fork
- : to split a Vault - binary approval or splitting/merging Vaults.
-
    - dink
-
    - : amount of collateral to exchange.
-
    - dart
-
    - : amount of stablecoin debt to exchange.
- *
- wish
- : check whether an address is allowed to modify another address's gem or dai balance.
-
    - hope
-
    - : enablewish
-
    - for a pair of addresses.
-
    - nope
-
    - : disablewish
-
    - for a pair of addresses.

- *
-

Note: art andArt represent normalized debt, i.e. a value that when multiplied by the correct rate gives the up-to-date, current stablecoin debt.

Accounting

- debt
- is the sum of alldai
- (the total quantity of dai issued).
- vice
- is the sum of allsin
- (the total quantity of system debt).
- Ilk.Art
- the sum of allart
- in theurn
- s for thatIlk
- .
- debt
- isvice
- plus the sum ofIlk.Art * Ilk.rate
- across allilks
- .
-

Collateral

- gem
- can always be transferred to any address by it's owner.
-

Dai

- dai
- can only move with the consent of it's owner.
- dai
- can always be transferred to any address by it's owner.
-

1. Mechanisms & Concepts

The core Vault, Dai, and collateral state is kept in theVat . TheVat contract has no external dependencies and maintains the central "Accounting Invariants" of Dai. The core principles that apply to thevat are as follows:

1. Dai cannot exist without collateral:
2.

3. Anilk

4. is a particular type of collateral.
5. Collateralgem
6. is assigned to users withslip
7. .
8. Collateralgem
9. is transferred between users withflux
10. .
11.

12. The Vault data structure is the Urn :

13. hasink

14.
    - encumbered collateral
15. hasart
16.
    - encumbered, normalized debt
17.

18. Similarly, a collateral is an Ilk :

18. Similarly, a collateral is an ilk .
19. hasArt
20.
    - encumbered, normalized debt
21. hasrate
22.
    - debt scaling factor (discussed further below)
23. hasspot
24.
    - price with safety margin
25. hasline
26.
    - debt ceiling
27. hasdust
28.
    - debt floor
29.

Note: Above, when using the term "encumbered", this refers to being "locked in a Vault".

Vault Management

- Vaults are managed viafrob(i, u, v, w, dink, dart)
- , which modifies the Vault of useru
- , usinggem
- from userv
- and creatingdai
- for userw
- .
- Vaults are confiscated viagrab(i, u, v, w, dink, dart)
- , which modifies the Vault of useru
- , givinggem
- to userv
- and creatingsin
- for userw
- .grab
- is the means by which Vaults are liquidated, transferring debt from the Vault to a userssin
- balance.
- Sin represents "seized" or "bad" debt and can be canceled out with an equal quantity of Dai usingheal(uint rad
- wheremsg.sender
- is used as the address for thedai
- andsin
- balances.
-
    - Note:
-
    - Only the Vow will ever havesin
-
    - , so only the Vow can successfully callheal
-
    - . This is because whenevergrab
-
    - andsuck
-
    - are called, the Vow's address is passed as the recipient ofsin
-
    - . Note that this is contingent on the current design and implementation of the system.
-
    - Note:
-
    - heal
-
    - can only be called with a positive number (uint) and willsub(dai[u])
-
    - along withsub
-
    - ing thesin
-

    - .
- *
- The quantitydai
- can be transferred between users withmove
- .
-

Rate Updates via fold(bytes32 ilk, address u, int rate)

An ilk'srate is the conversion factor between any normalized debt (art) drawn against it and the present value of that debt with accrued fees. Therate parameter tofold is actually the change in theIlk.rate value, i.e. a difference of scaling factors (new - old). It is a signed integer, and hence current account values may increase or decrease. The quantityIlk.Art*rate is added to thedai balance of the addressu (representing an increase or decrease in system surplus); the debt balances of all Vaults collateralized with the specifiedIlk are updated implicitly via the addition ofrate toIlk.rate .

For more information on Rates and System Stabilization, see the Rates Module and System Stabilizer Module documentation below:

- Rates Module
- System Stabilizer
-

1. Gotchas

The methods in theVat are written to be as generic as possible and as such have interfaces that can be quite verbose. Care should be taken that you have not mixed the order of parameters.

Any module that isauth ed against theVat has full root access, and can therefore steal all collateral in the system. This means that the addition of a new collateral type (and associated adapter) carries considerable risk.

1. Failure Modes

Coding Error

A bug in theVat could be catastrophic and could lead to the loss (or locking) of all Dai and Collateral in the system. It could become impossible to modify Vault's or to transfer Dai. Auctions could cease to function. Shutdown could fail.

Feeds

TheVat relies upon a set of trusted oracles to provide price data. Should these price feeds fail, it would become possible for unbacked Dai to be minted, or safe Vaults could be unfairly liquidated.

Governance

Governance can authorize new modules against theVat . This allows them to steal collateral (slip) or mint unbacked Dai (suck / addition of worthless collateral types). Should the cryptoeconomic protections that make doing so prohibitively expensive fail, the system may be vulnerable and left open for bad actors to drain collateral.

Adapters

TheVat relies on external Adapter contracts to ensure that the collateral balances in theVat represent real external collateral balances. Adapter contracts are authorized to make arbitrary modifications to all collateral balances. A faulty collateral adapter could result in the loss of all collateral in the system.

Previous Core Module Next Spot - Detailed Documentation Last updated2 years ago On this page *1. Introduction (Summary) * 2. Contract Details * Glossary (Vat - Vault Engine) * 3. Mechanisms & Concepts * Vault Management * Rate Updates via fold(bytes32 ilk, address u, int rate) * 4. Gotchas * 5. Failure Modes

Export as PDF