Jug - Detailed Documentation

Accumulation of Stability Fees for Collateral Types * Contract Name: * Jug * Type/Category: * DSS —> Rates Module * Associated MCD System Diagram * Contract Source * Etherscan *

1. Introduction

Summary

The primary function of the Jug smart contract is to accumulate stability fees for a particular collateral type whenever itsdrip() method is called. This effectively updates the accumulated debt for all Vaults of that collateral type as well as the total accumulated debt as tracked by the Vat (global) and the amount of Dai surplus (represented as the amount of Dai owned by theVow).

?

1. Contract Details

Structs

Ilk : contains twouint256 values—duty , the collateral-specific risk premium, andrho , the timestamp of the last fee update

VatLike : mock contract to make Vat interfaces callable from code without an explicit dependency on the Vat contract itself

Storage Layout

wards :mapping(address => uint) that indicates which addresses may call administrative functions

ilks :mapping (bytes32 => Ilk) that stores anIlk struct for each collateral type

vat : aVatLike that points the the system'sVat contract

vow : theaddress of the Vow contract

base : auint256 that specifies a fee applying to all collateral types

Public Methods

Administrative Methods

These methods requirewards[msg.sender] == 1 (i.e. only authorized users may call them).

rely /deny : add or remove authorized users (via modifications to thewards mapping)

init(bytes32) : start stability fee collection for a particular collateral type

file(bytes32, bytes32, uint) : setduty for a particular collateral type

file(bytes32, data) : set thebase value

file(bytes32, address) : set thevow value

Fee Collection Methods

drip(bytes32) : collect stability fees for a given collateral type

1. Key Mechanisms & Concepts

drip

drip(bytes32 ilk) performs stability fee collection for a specific collateral type when it is called (note that it is a public function and may be called by anyone).drip does essentially three things:

1. calculates the change in the rate parameter for the collateral type specified byilk
2. based on the time elapsed since the last update and the current instantaneous rate (base + duty
3.);
4. callsVat.fold
5. to update the collateral'srate
6. , total tracked debt, and Vow surplus;
7. updatesilks[ilk].rho

8. to be equal to the current timestamp.
9.

The change in the rate is calculated as:

! Δ r a t e = (b a s e + d u t y) n o w – r h o · r a t e – r a t e \Delta rate = (base+duty)^{now-rho} \cdot rate- rate where "now" represents the current time, "rate" isVat.ilks[ilk].rate , "base" isJug.base , "rho" isJug.ilks[ilk].rho , and "duty" isJug.ilks[ilk].duty . The function reverts if any sub-calculation results in under- or overflow. Refer to the Vat documentation for more detail onfold .

rpow

rpow(uint x, uint n, uint b) , used for exponentiation indrip , is a fixed-point arithmetic function that raisesx to the powern . It is implemented in Solidity assembly as a repeated squaring algorithm.x and the returned value are to be interpreted as fixed-point integers with scaling factorb . For example, ifb == 100 , this specifies two decimal digits of precision and the normal decimal value 2.1 would be represented as 210;rpow(210, 2, 100) returns 441 (the two-decimal digit fixed-point representation of 2.1^2 = 4.41). In the current implementation, 10^27 is passed forb , makingx and therpow result both of typeray in standard MCD fixed-point terminology.rpow 's formal invariants include "no overflow" as well as constraints on gas usage.

Parameters Can Only Be Set By Governance

Jug stores some sensitive parameters, particularly the base rate and collateral-specific risk premiums that determine the overall stability fee rate for each collateral type. Its built-in authorization mechanisms need to allow only authorized MakerDAO governance contracts/actors to set these values. See "Failure Modes" for a description of what can go wrong if parameters are set to unsafe values.

1. Gotchas (Potential Sources of User Error)

Ilk Initialization

init(bytes32 ilk) must called when a new collateral is added (settingduty viafile() is not sufficient)—otherwiserho will be uninitialized and fees will accumulate based on a start date of January 1st, 1970 (start of Unix epoch).

base + Ilk.duty imbalance indrip()

A call todrip(bytes32 ilk) will add thebase rate to theIlk.duty rate. The rate is a calculated compounded rate, sorate(base + duty) != rate(base) + rate(duty) . This means that if base is set, the duty will need to be set factoring the existing compounding factor in base, otherwise the result will be outside of the rate tolerance. Updates to thebase value will require all of theilks to be updated as well.

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

Tragedy of the Commons

Ifdrip() is called very infrequently for some collateral types (due, for example, to low overall system usage or extremely stable collateral types that have essentially zero liquidation risk), then the system will fail to collect fees on Vaults opened and closed betweendrip() calls. As the system achieves scale, this becomes less of a concern, as both Keepers and MKR holders are have an incentive to regularly call drip (the former to trigger liquidation auctions, the latter to ensure that surplus accumulates to decrease MKR supply); however, a hypothetical asset with very low volatility yet high risk premium might still see infrequent drip calls at scale (there is not at present a real-world example of this—the most realistic possibility isbase being large, elevating rates for all collateral types).

Malicious or Careless Parameter Setting

Various parameters of Jug may be set to values that damage the system. While this can occur by accident, the greatest concern is malicious attacks, especially by an entity that somehow becomes authorized to make calls directly to Jug's administrative methods, bypassing governance. Settingduty (for at least one ilk) orbase too low can lead to Dai oversupply; setting either one too high can trigger excess liquidations and therefore unjust loss of collateral. Setting a value forvow other than the true Vow's address can cause surplus to be lost or stolen.

Previous Pot - Detailed Documentation Next Proxy Module Last updated3 years ago On this page *1. Introduction * Summary * 2. Contract Details * Structs * Storage Layout * Public Methods * 3. Key Mechanisms & Concepts * drip * rpow * Parameters Can Only Be Set By Governance * 4. Gotchas (Potential Sources of User Error) * Ilk Initialization * base + Ilk.duty imbalance in drip() * 5. Failure Modes (Bounds on Operating Conditions & External Risk Factors) * Tragedy of the Commons * Malicious or Careless Parameter Setting

Export as PDF