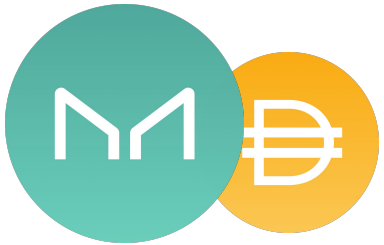
Some sections (which are highlighted in red) are still a WIP
Updated Dec 2nd, 2019



Maker Protocol 101

Kenton, Wouter, Soren, Tom, and Chris B.

Maker Foundation

Please submit errors to kenton@makerdao.com or @Kenton on Rocket Chat

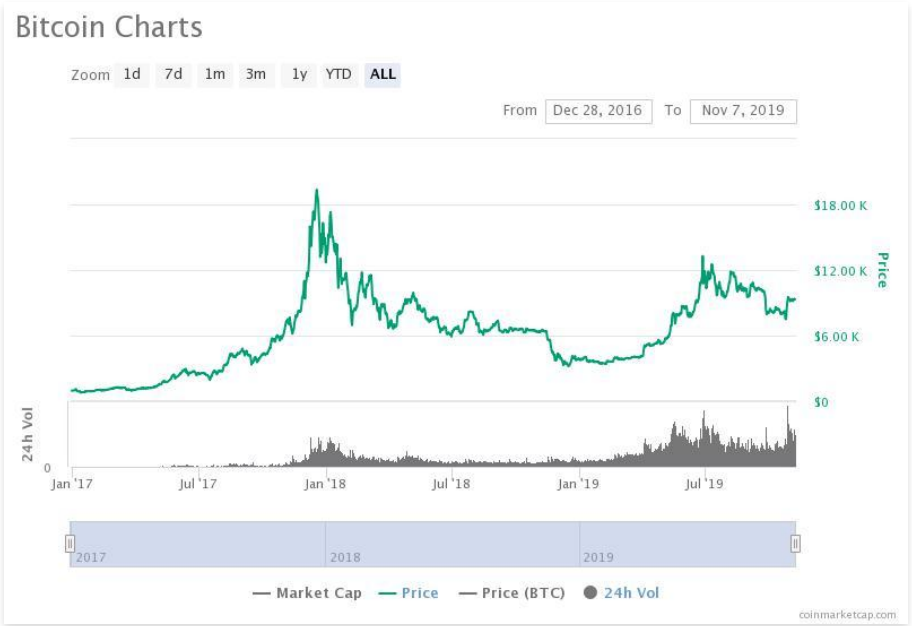
Sections

1. **30 seconds**
    a. Problem and Solution
    b. Dai
2. **3-5 minutes**
    a. Economics
    b. Vault Mechanics
3. **60 minutes**
    a. Smart contract Modules
4. **90 minutes**
    a. Advanced Concepts

Problem: Instability

The ultimate gauge of capital is denominated in the Global Currency, which is USD.

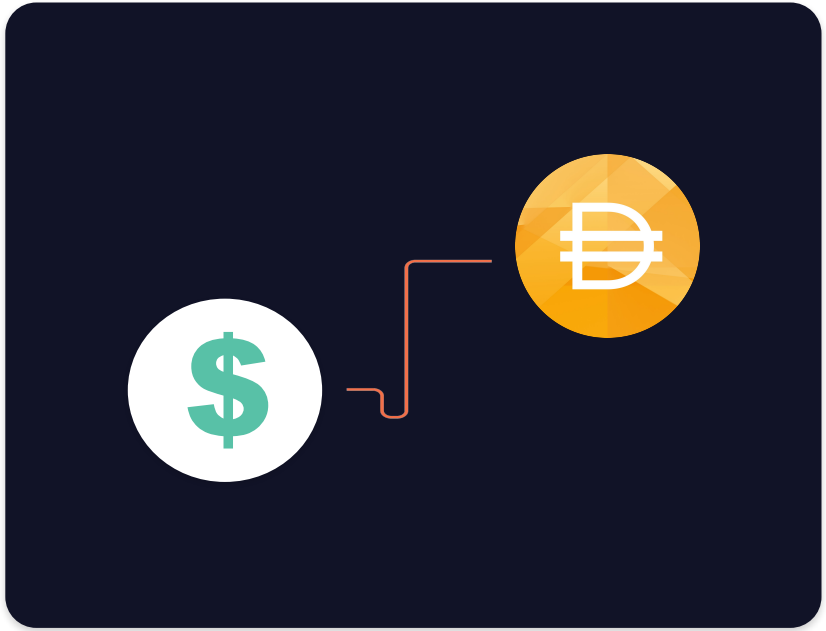
Is Bitcoin stable relative to the Global Currency?



Maker’s Flagship Product

Dai - Stablecoin

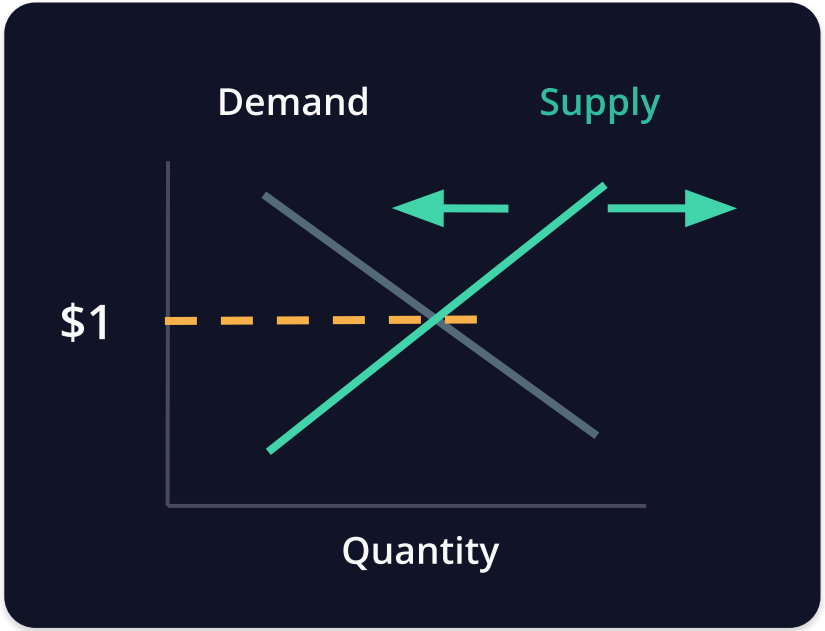
- Dai soft-pegged to USD
- Basic user
- Fully backed by collateral



Dai - Economics

How does it keep its peg?

- Demand curve can shift due to market conditions, confidence of Dai holders, etc
- Supply curve is shifted through a permissionless credit factory on Ethereum
- Any actor can vary the supply of Dai through a Maker Vault
- The system was built so that these actors are incentivized to shift the supply curve to ensure that the price is close as possible to $1



Dai - Vault

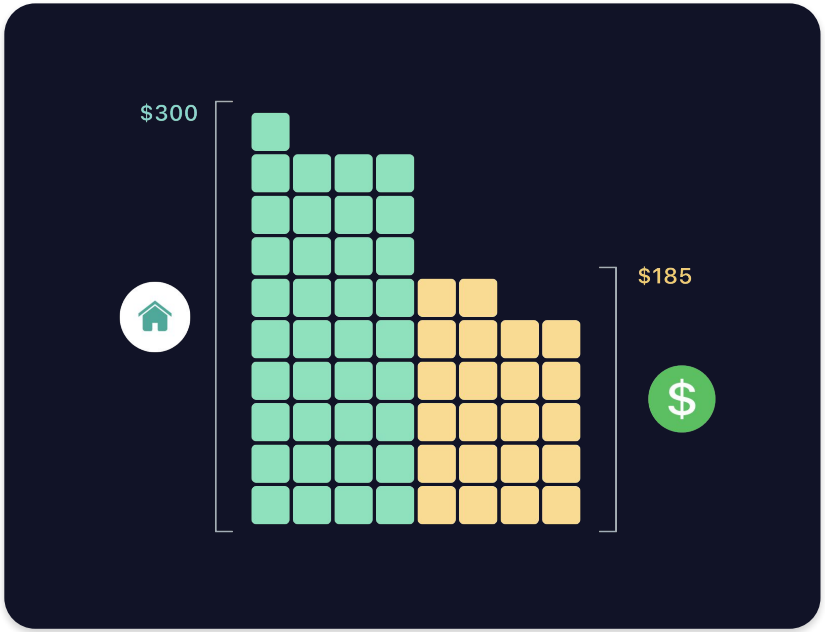
Maker Vault

- Borrow Dai through locking up crypto assets as collateral
- Repay Dai + fee to retrieve collateral
- Safe, over-collateralized Vault >

Dai - Vault

Analogous to a Mortgage

- Bank gives you a loan by “locking” ownership rights with them
- Repay debt to “free” the bank’s ownership of the house



Dai - Vault

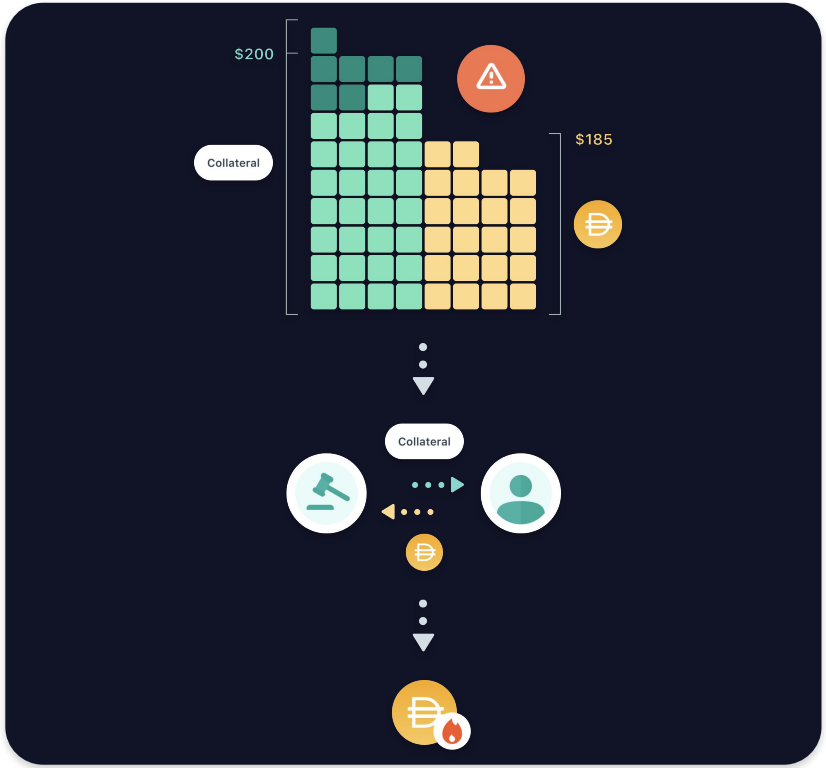
Liquidations

1. A Vault is automatically liquidated if the collateral value (in USD) falls too low

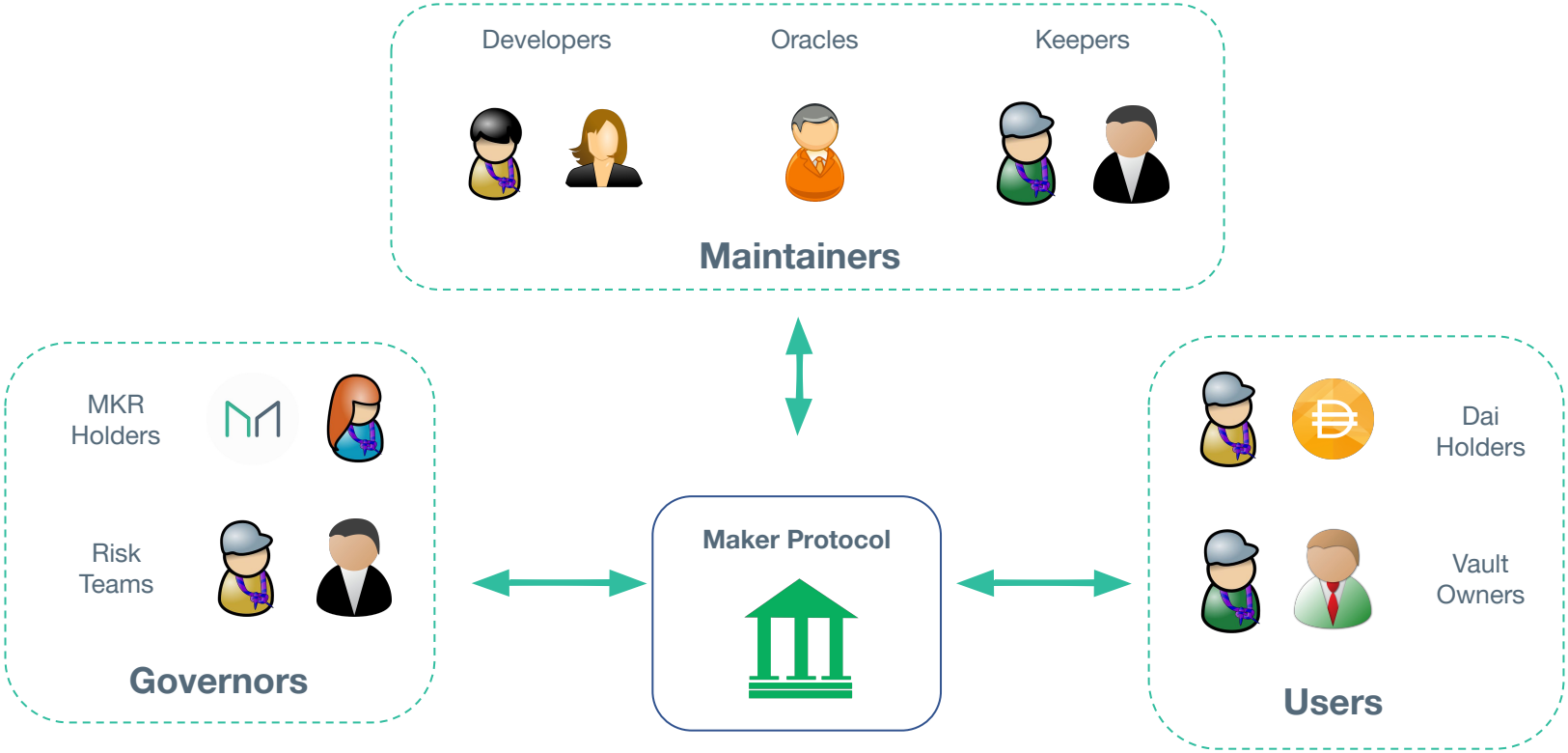
2. Part of the collateral is auctioned off by the Protocol to cover the outstanding debt + penalty fee

3. Dai is then burned by the Protocol to decrease the supply

   Vault owner receives the leftover collateral



System Interaction Diagram



System Interaction Actors

User

- Dai Holders
  - Basic - No additional knowledge to own Dai
  - Stability Seekers; Consumer; Businesses
- Vault Owners
  - Intermediate - Some knowledge
  - Risk Seekers; Speculators; Borrowers

Governor

- MKR Holders
  - Advanced - Extensive knowledge
  - Monitor, partake, and vote on upgrades/changes in the Maker Protocol
- Risk Teams
  - Advanced - Extensive knowledge
  - Collect/compile relevant data and develop risk models, assessed by MKR Holders

System Interaction Actors

Maintainer

- Developers
  - Basic - No additional knowledge to own Dai
  - Stability Seekers; Consumer; Businesses
- Oracles
  - Intermediate - Some knowledge
  - Risk Seekers; Speculators; Borrowers
- Keepers
  - Intermediate - Some knowledge
  - Risk Seekers; Speculators; Borrowers

Anyone with the required knowledge can freely participate in any role

Any one person (or service) can have multiple roles

System Lines of Defense

At any point, the system must have more value in collateral than value of Dai supply.

The following mechanisms help maintain system solvency:

1. Supply and Demand	Supply and demand of Vaults (and thus Dai) is influenced by the Stability Fees, Dai Savings Rate, and Debt Ceiling adjustments.
2. Liquidation	Any time the collateral value of a Vault gets closer to its debt, it becomes “risky-er”. The system liquidates Vaults that get too risky.

System Lines of Defense

3. MKR Minting/Burning

If MKR holders govern the Maker Protocol successfully, the Protocol will accrue Surplus Dai as Dai holders pay Stability Fees. On the other hand, if liquidations are inadequate, then the Protocol will accrue Bad Debt. Once this Surplus Dai / Bad Debt amount hits a threshold, as voted by MKR holders, then the Protocol will discharge Surplus Dai / Bad Debt through the Flapper / Flopper smart contract by buying and burning / minting and selling MKR, respectively.

4. Emergency Shutdown

This is a process that is used as a last resort in cases of extreme market irrationality, attacks, or coordinated upgrades. Emergency Shutdown gracefully settles the Maker Platform while ensuring that all users, both Dai holders and Vault users, receive the net value of assets they are entitled to.

Smart Contract Modules

Table of Contents

- Sai vs Dai
- Vocabulary
- Smart Contract Modules

Sai vs Dai

Single Collateral DAI

- Single collateral type backing Dai
- Collateral is ETH
- Less diversified
- Liquidations occur at a fixed discount to the current collateral price
- MKR used to pay stability fee
- Decentralized governance through MKR voting rights

Multi Collateral DAI

- Multi. collateral types backing Dai
- Collateral can be any approved asset
- More diversified and stable
- Liquidations are handled using Auctions
- Dai used to pay stability fee
- Decentralized governance through MKR voting rights

Sai vs Dai - Terminology

Single Collateral DAI

- Dai Credit System
- CDP
- Boom / Bust Spread

Multi Collateral DAI

- Maker Protocol
- (Maker) Vault
- Discount / Premium Spread

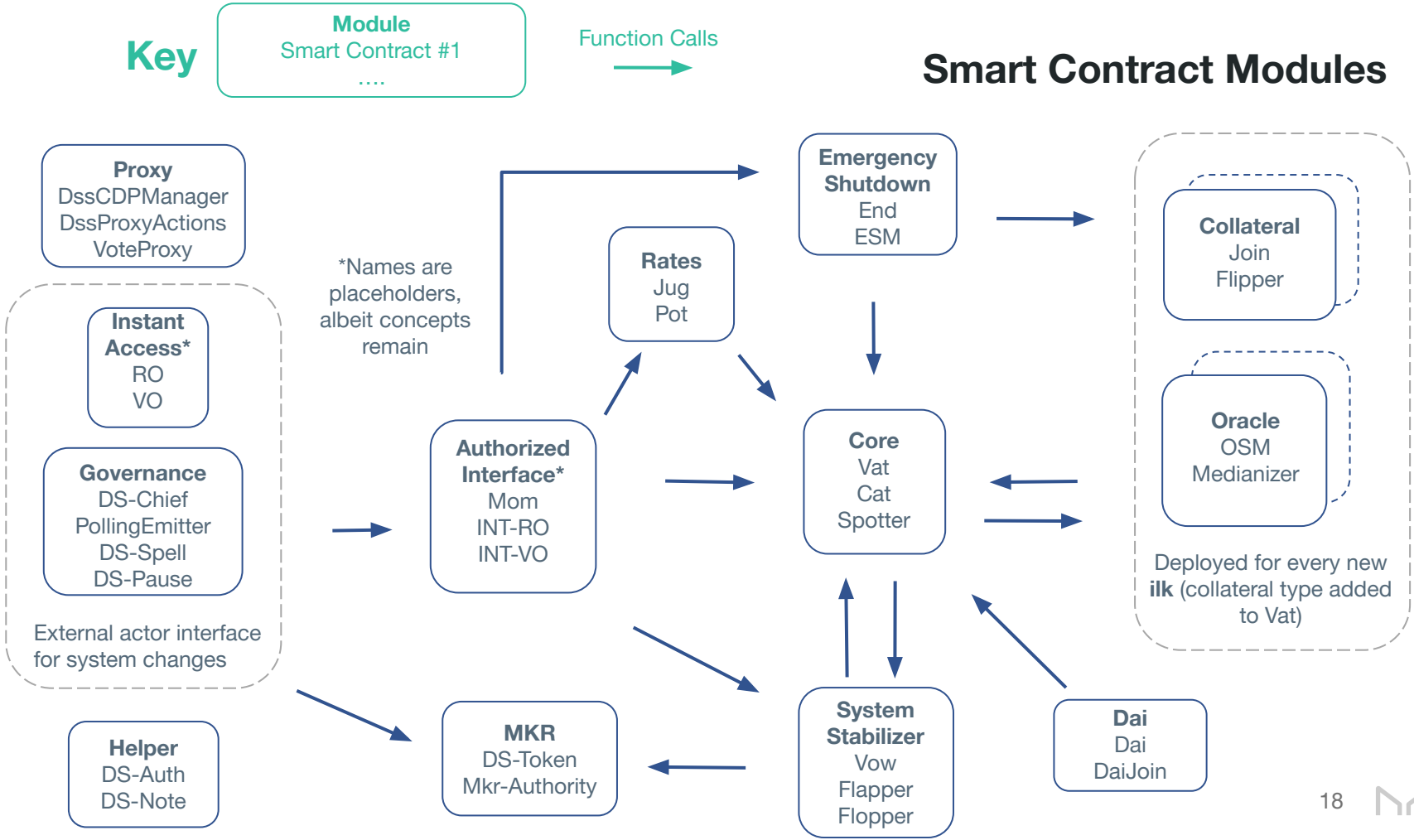
Quick Vocab - Rates

- **Risk Premium Rate** - This rate is used to calculate the risk premium fee that accrues on debt in a Vault. A unique Risk Premium Rate is assigned to each collateral type. (e.g. 2.5%/year for Collateral A, 3.5%/year for Collateral B, etc)
- **Base Rate** - This rate is used to calculate the base fee that accrues on debt in a Vault. A system wide Base Rate is assigned to all collateral types. (e.g. 0.5%/year for the Maker Protocol)
- **Stability Rate** = *Risk Premium Rate* + *Base Rate*. This rate is used to calculate the **Stability Fee**.
- **Savings Rate** - This rate is used to calculate the dai earned that accrues on Dai locked in the savings contract. A system wide Savings Rate is assigned to all Savings Dai. (e.g. 1%/year for Savings Dai)

All rates accrue regularly on a per second basis.

https://github.com/makerdao/dss/wiki/Glossary

Smart Contract Modules



Motivations for concise names

To sidestep terminological debates; for example, whether to say *rate of target price change* or *target rate*

To decouple financial and technical vocabularies; we can more flexibly improve one without affecting the other.

To discuss the system formally; this ability, with the financial interpretation partly suspended, has suggested insights that would have been harder to think of inside the normal language.

To better formalize the implementation; the precise and distinctive language makes the structure and logic of the implementation more apparent and easier to formalize.

To decrease verbosity; concise names make the code less verbose and the concepts easier to handle on paper, whiteboard, etc.

Quick Vocab - general

Risk Parameters - system variables adjusted through MKR governance to control various types of risk.

Important parameters for each collateral type:

- **Stability fee** - a fee that continuously accrues on debt in a Vault (e.g. 2.5% per year)
- **Debt ceiling** - max amount of Dai generated for a given collateral type (e.g. 10 million Dai)
- **Liquidation ratio** - minimum ratio of collateral value to debt per Vault (e.g. 150%)

ilk - collateral type; each has its own set of risk parameters

urn - vault; an ethereum address can control one *urn per collateral type*

gem - unlocked collateral; gem is collateral that is not yet locked in a Vault but still recorded in the system

sin - system debt unit; a debt balance that is tracked during liquidation process

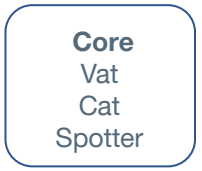
dai - stablecoin; a good debt token

https://github.com/makerdao/dss/wiki/Glossary

Core Module - what

The core module contains the state of the Maker Protocol and its central mechanisms while in normal operation.

Components



- **Vat** - **The single source of truth for the Maker Protocol.** It contains the accounting system of the core Vault, Internal Dai balances, and collateral state. The Vat has no external dependencies and maintains the central "Accounting Invariants" of the Maker Protocol. It houses the public interface for Vault management, allowing urn (Vault) owners to adjust their Vault state balances. It also contains the public interface for Vault fungibility, allowing urn (Vault) owners to transfer, split, and merge Vaults. Excluding these interfaces, the Vat is accessed through *trusted smart contract modules*.
- **Cat** - Public interface for **confiscating unsafe urns** (Vaults) and **processing seized collateral** via their respective flip (collateral) auction. With large Vaults, partial confiscations of a fixed collateral size will be processed until the urn becomes safe again.
- **Spotter** - Allows external actors to **update the price feed** in Vat for a given Ilk (collateral type).

Core Module - why Vat

Components

- **Vat** - def. “a large tank or tub used to hold liquid”

  The Vat holds the fundamental primitives of the Maker Protocol:

  - Database - Risk parameters as well as Dai, Sin, collateral, and debt balances
  - Accounting System - Basic accounting operations to update the Database
  - Vault Management - Adjustment of locked collateral and debt position (Dai creation)
  - Vault Fungibility - Ability to transfer, split and merge Vaults

  The Vat and these primitives are designed to be non-upgradable or replaceable.

  Other components of the system, such as auctions, oracles, and rate accumulators, are subject to future development. Their business logic has been contained within their own smart contract modules as an upgradable interface between the user and the Vat. Since these subsystems have access to more complex operations within the Vat, module upgrades are voted in by MKR holders. Through Vault management/fungibility, direct access to the Vat provides a strong guarantee to users that the basic semantics/interface aren’t going to change.

https://github.com/makerdao/dss/blob/master/src/vat.sol

Core Module - why Cat

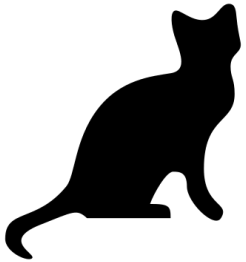
Components Continued

- **Cat**

  The Cat **“bites” Vaults that are too risky**.

  The generator of Dai is a Vault owner and the “borrower”, while the smart contract system is the “generator”. To increase the borrower’s confidence in the minted value lent out, the generator requires all minted Dai to be fully backed by an asset with value that is proven in free markets. If the value of the underlying asset dips below the required amount, the collateralization ratio (USD value of asset / USD value of Dai debt) decreases. To increase this ratio and prevent insolvency of the system, the generator takes (via Cat) the collateral and sells it for Dai in an auction (Flipper).

  Remember: 1 Dai must always be backed by more than 1 USD worth of assets.



https://github.com/makerdao/dss/blob/master/src/cat.sol

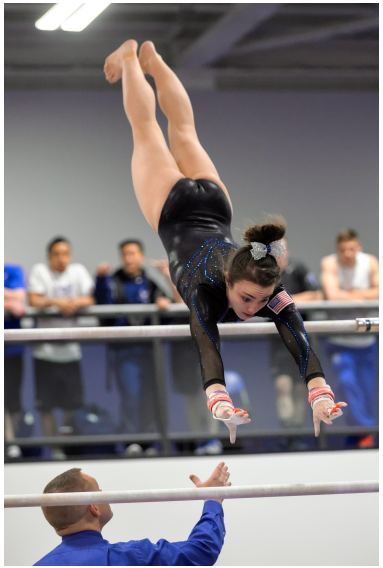
Core Module - why Spotter

Components Continued

- **Spotter** - def. “Gymnastics Spotting or to “spot” someone means to physically assist them in safely completing a skill”

  The Maker Protocol requires real time information about the market price of the assets used as collateral in Vaults. Ultimately, this market price determines the amount of Dai that can be minted, as well as the grab condition for Vault liquidations. The oracle module handles how markets prices are recorded on the blockchain.

  The Spotter is simply an **interface contract** where external actors pull the **current market price** from the Oracle module for the **specified collateral type**. The Vat reads the market price from the spotter.



https://github.com/makerdao/dss/blob/master/src/spot.sol

Collateral Module - what

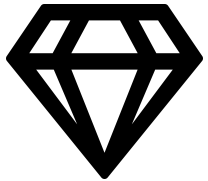
The collateral module is deployed for every new ilk (collateral type) added to Vat. It contains all adapters and auction contract for one collateral type. All token behavior is *abstracted behind* these adapters.

Components

- **Join** - The Join adapter is used to **deposit/withdraw unlocked collateral** into the Vat. Currently, GemJoin is only compatible with standard ERC20 tokens, but eventually there will be various types of Join adapters that are compatible with different Token Standards.
- **Flipper** - **Collateral auction house.** Each gem auction is unique and linked to a previously bitten urn (Vault). Investors bid with increasing amounts of DAI for a fixed GEM amount. When the DAI balance deficit is covered, bidders continue to bid for a decreasing gem size until the auction is complete. Remaining GEM is returned to the Vault owner.

Collateral Module - why

Components



- **Join** - def. of Gem - “a precious or semiprecious stone”

  To retain the security of the system, only trusted smart contracts can add/remove value to/from the Vat. A Join adapter is a trusted smart contract that is used to **deposit unlocked collateral** (gem) within the Vat. The location of collateral deposited/locked in Vaults is in the the appropriate Join adapter.

- **Flipper** - def. of Flip - “turn over or cause to turn over with a sudden sharp movement.”

  The purpose of the gem auction house is to decrease the market risk of collateral backing Dai. It sells an amount of seized gem to **purchase and burn Dai,** which will **increase the collateralization ratio** of the system, away from insolvency.

  Priorities for the Flipper:

  1. “Tend” phase: Cover the amount of total debt (minted Dai + accrued fees) of the Vault
  2. “Dent” phase: Return as much collateral back to the Vault owner as possible

https://github.com/makerdao/dss/blob/master/src/join.sol
https://github.com/makerdao/dss/blob/master/src/flip.sol

Dai Module - what

Fundamentally, 'Dai' is any token that the core system considers equal in value to its internal debt unit.

The dai module contains the dai token representation and all adapters thereof.

Components

- **Dai** - An extension from DS-Token and standard ERC20 token interface. Contains the database of Dai token owners, transfer, approval and supply logic.
- **DaiJoin** - DaiJoin is an adapter where all Dai tokens are created. The Vault owner interacts with DaiJoin to mint the Dai tokens that has been allocated for them in the Vat as well as burn Dai Tokens + fees accrued against their Vault.

Dai Module - why

Dappsys

Components

- **Dai** - Dai is an extension of DS-Token, a contract within DappSys, a safe, simple, flexible library for smart-contract systems.

  DSToken is an implementation that supports the **ERC20 Standard, but with a few additions** that complement the design of the Maker Protocol:

  - *Addition of mint and burn functions (with proper authorization)* -> to control token supply
  - *'push', 'pull' and 'move' aliases for transferFrom operations* -> improves readability
  - *Binary allowance approval* -> lower gas and higher security

  Dai contains a novel permit function that allows users to **give allowance without needing ETH**

- **DaiJoin**

  DaiJoin is a trusted smart contract that is used to deposit Dai into the Vat. All minting and burning of Dai tokens happens in DaiJoin (think "United States Mint").

https://github.com/makerdao/dss/blob/master/src/dai.sol
https://github.com/makerdao/dss/blob/master/src/join.sol

System Stabilizer Module - what

When the value of the collateral backing Dai drops below the liquidation level, then the stability of the system is at risk. The system stabilizer module sets up incentives for Keepers (incentivized external actors) to step in, push the system back to a safe state, and earn profits.

System Stabilizer
Vow
Flapper
Flopper

Components

- **Vow** - The Vow represents the **Maker Protocol's balance**, as the recipient of both system surplus and system debt. Its function is to cover deficits via debt auctions and discharge surpluses via surplus auctions.
- **Flopper** - Debt Auction house. Debt auctions are used to satisfy the Vow's debt by auctioning off MKR for a fixed amount of internal Dai. **Bidders compete with decreasing "amount requests" of MKR**. After auction settlement, the Flopper sends the received internal Dai to the Vow to cancel out its debt. The Flopper mints the MKR for the winning bidder.
- **Flapper** - Surplus Auction house. Surplus auctions are used to liquidate the Vow's surplus by auctioning off a fixed size of internal Dai for MKR. **Bidders compete with increasing amounts of MKR.** After auction settlement, the Flapper burns the winning MKR bid and sends the internal Dai to the winning bidder.

System Stabilizer Module - why

Components

- **Vow** - def. “a solemn promise.”
  The Maker Protocol deviates from equilibrium when it receives system debt and system surplus through collateral auctions, Dai Savings and Vault stability fee accumulation. The Vow houses the business logic to kick off debt and surplus auctions, which correct the system’s monetary imbalances.

  System debt: **When Vaults are bitten**, their debt is taken on by the Vow as Sin, the system debt unit, and placed in the Sin queue. If this Sin is not covered by a flip auction within some wait time, the Sin “matures” and is now considered bad debt to the Vow. This bad debt can be covered through a debt auction when it exceeds a minimum value (the `lot` size). The source of **Dai Savings Accumulation** comes from increasing Sin (system debt) in the Vow.

  System surplus: **Stability fee accumulation** occurs in the form of additional internal Dai to the Vow. This surplus is then discharged through surplus auctions.

https://github.com/makerdao/dss/blob/master/src/vow.sol

System Stabilizer Module - why pt. 2

Components continued

- **Flopper** - def. of Flop - “be completely unsuccessful; fail totally”

  The purpose of the debt auction is to cover the system deficit, which is represented by Sin. It sells an amount of minted MKR and purchases Dai to be canceled 1-to-1 with Sin.

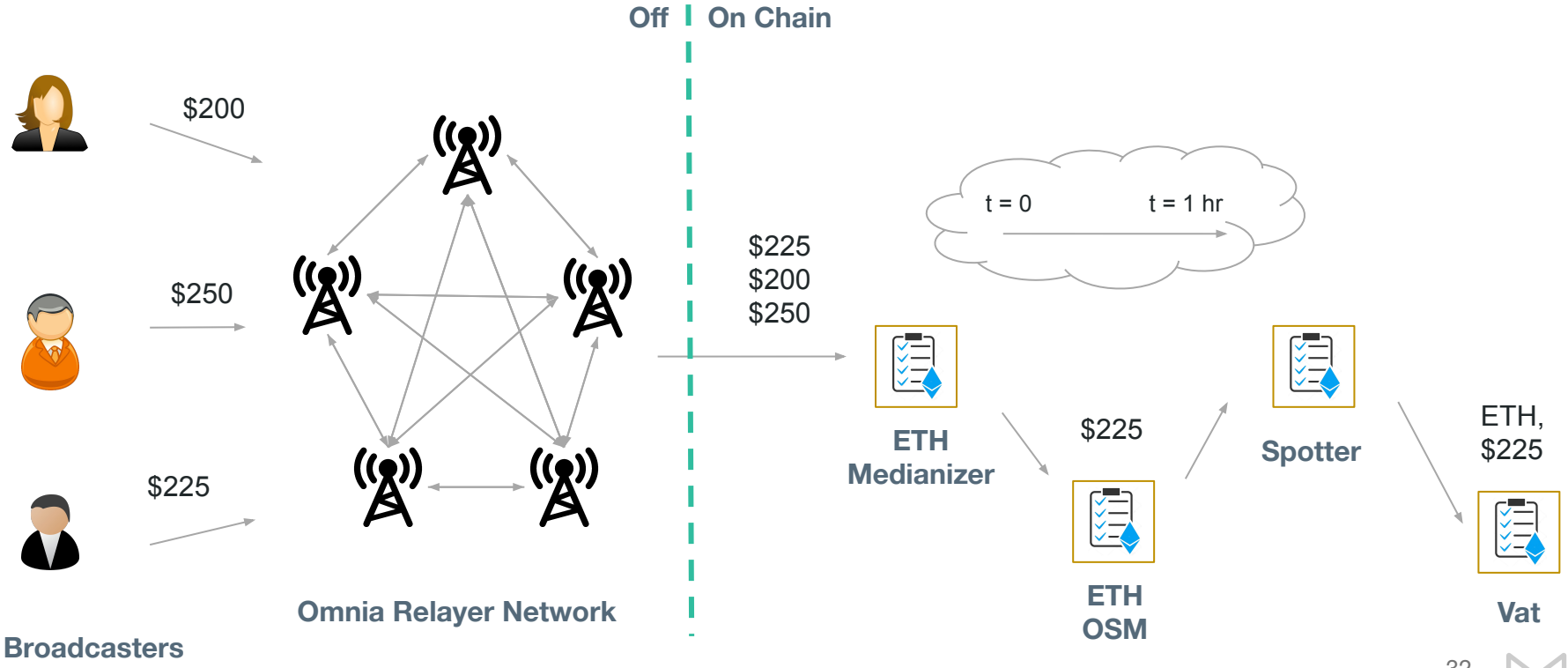
  Priorities for the Flopper:
  1. Raise an amount of Dai equivalent to the amount of bad debt **as fast as possible**
  2. Minimize the amount of MKR inflation

- **Flapper** - def of Flap - “(of a bird) move (its wings) up and down when flying or preparing to fly”

  The purpose of the surplus auction is to release Dai surplus from the Vow. It sells a fixed amount of Dai to purchase and burn a bid amount of MKR.

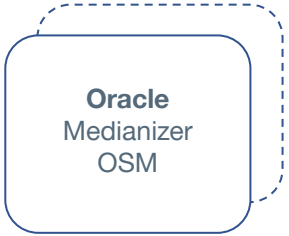
https://github.com/makerdao/dss/blob/master/src/flap.sol
https://github.com/makerdao/dss/blob/master/src/flop.sol

Oracle Module - what



Oracle Module - (what & why) pt. 2

The value of collateral in a Vault is derived from its global, free market USD price. An oracle module is deployed for each collateral type. It feeds price data for a corresponding collateral type to the Vat. Whitelisted addresses broadcast price updates off-chain, which are fed into a medianizer before being pulled into the OSM. The Spotter reads from the OSM.



Components

- **Medianizer** - For a specific Ilk, the Medianizer returns the median value of several price feeds, fed from the off-chain Omnia Relayer Network. A median value is determined to mitigate the variability in single data points.
- **OSM** (Oracle Security Module) - Authorized users are allowed to set a value after some duration of time (e.g. one hour). To protect the system from an attacker who gains control of a majority of the oracles, the OSM imposes a **1 hour delay on price feeds**, leaving enough time for the MKR governance community to analyze the data and react.

https://github.com/makerdao/medianizer
https://github.com/makerdao/osm

Mkr Module - what

The MKR module contains the MKR token representation and its custom authority

MKR
DS-Token
Mkr-Authority

Components

- **DS-Token** - An implementation supporting the ERC20 Standard; part of the DappSys (DS) library. Contains database of MKR owners, transfer and supply logic.
- **Mkr-Authority** - Custom authority contract for allowing **Maker Governance to govern the MKR** token contract. Mkr-Authority is controlled by the DSPauseProxy, which is exclusively owned by DS-Pause. Thus, DS-Pause has indirect permission to give authority to other contracts to call certain functions on the MKR token contract. In practice, this is how the Flopper contract has access to call the `mint()` function during the Protocol's Debt Auctions. Such a structure allows governance proposals voted in on the Chief to make arbitrary changes to the MKR token and its permissions, subject to a delay.

https://etherscan.io/address/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2#code
https://github.com/makerdao/mkr-authority

Mkr Module - why

The MKR module contains the MKR token representation and its custom authority

As a **utility token**: As Dai stability fees earned on Vaults accrue within the Maker Protocol, MKR holders may vote to enable the Flapper auction house to sell Dai surplus for MKR. Once the auction is complete, the Maker Protocol burns the MKR.

As a **governance token**: MKR is used by MKR holders to vote for the risk management and business logic of the Maker Protocol. Tokens are a simple representation of voting power.

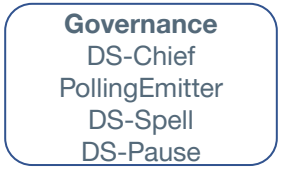
As a **recapitalization resource**: In narrow circumstances, MKR can be autonomously minted by the Flopper auction house and sold for DAI, which is used to recapitalize the Maker Protocol in times of insolvency.

https://medium.com/makerdao/what-is-mkr-e6915d5ca1b3

MKR
DS-Token
Mkr-Authority

Governance Module - what

The Governance Module contains the contracts that facilitate MKR voting, proposal execution, and voting security of the Maker Protocol.



Components

- **DS-Chief** - A basic voting contract that grants root access of the Maker Protocol to an elected “Chief” (address). Through Approval Voting, voters **lock up their MKR and vote** with a weight relative to the outstanding supply of MKR. Spells (proposals) are a type of Proposal Object and are submitted to DS-Chief as Executive Proposals, which can make a change to the protocol (adjusting risk parameters, upgrade adapters, add new collateral types, etc). Anyone can create a Spell, and MKR holders can vote on bundles of Spells, called Slates. At any point, **the Spell (proposal) with the most approval is the elected “Chief”**, has access to and can configure the Maker Protocol through Mom, the Admin interface contract for Maker Governance.

- **PollingEmitter** - Also known as the symbolic voting contract, the PollingEmitter is a **lightweight contract** that is used to **vote on Maker Governance polls**. Both the polls and the votes thereof are emitted events. The poll event contains the poll’s start date, end date, and a hash of its details, rules and metadata. The vote event contains the poll id and voter address. Anyone can create a poll and cast votes. Votes are tallied off-chain by reading the amount of MKR owned by the address in DS-Chief, held directly by the address, as well as any held in the vote proxy associated with the voter address. The checkpoint tally will be regularly performed during the poll, but the final tally will be at the block specified by the poll’s end date.

https://github.com/dapphub/ds-chief
https://github.com/makerdao/symbolic-voting

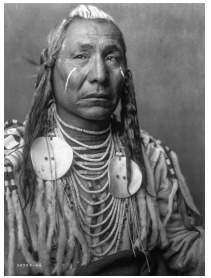
Governance Module - why

Components

- **DS-Chief** - def. of Chief - “a leader or ruler of a people or clan”

  DS-Chief is the first iteration of an on-chain tool for Maker Governance. It simply allows MKR holders to vote with a weight relative to their proportional holdings. **Votes are casted towards Proposal Objects**, which are ethereum addresses that can represent:

  - A Spell (contract with one function that does one action, one time)
  - A MegaSpell (contract with one function that does multiple actions, one time)
  - Multi-Signature Contract
  - Externally owned account, etc



  This governance mechanism employs an ACL (access control list) approach, where there is a single owner that has permission to call protected functions on Mom (Admin. interface for the Maker Protocol). **At any time, this single owner is the Proposal Object that has the most votes/approval in DS-Chief**; it could change during every executive voting period, which is facilitated through off-chain coordination. Within this period, MKR holders are encouraged to vote for a Slate (bundle of Spells) that includes the previous Spell. This secures the election of the old “Chief” until enough MKR approves the new Proposal Object. In other words, this prevents the chance of an unintended Proposal Object from being elected “Chief” during the approval phase of the new Proposal Object.

https://github.com/dapphub/ds-chief

Governance Module - why

Components

- **PollingEmitter** - def. of Emitter - “a machine, device, etc., that emits something”

  In the past, DS-Chief was used to gauge MKR holder sentiment through governance proposals (polls) and votes. This had profound limitations: an inability to run multiple concurrent polls and unwanted cross-contamination between polls and executive votes. To decouple governance and executive votes, the PollingEmitter contract was created to facilitate Maker Governance Polling. It has the following characteristics: censorship resistance, cryptographic verifiability, on-chain audit-ability, and minimal on-chain computation/ gas efficiency.

  All polls and votes are simply **events emitted by this contract**, and **all tallying is done off-chain**. The tradeoff is that, without state, other smart contracts can't query it for information. However, because symbolic votes don't need to be binding on-chain, it was decided that whatever could be pushed to social layer, should be. In brief, the idea here is that voters signal their intent with on-chain events, then services and community members can, using the rules defined in each poll, tally the polls and imbue them with meaning off-chain (e.g. by summing up the amount of MKR voting for each poll option at a certain block).

https://github.com/makerdao/symbolic-voting

Governance Module - (what and why) pt. 2

Components cont.

- **DS-Spell** - def. of Spell - “a form of words used as a magical charm or incantation.”

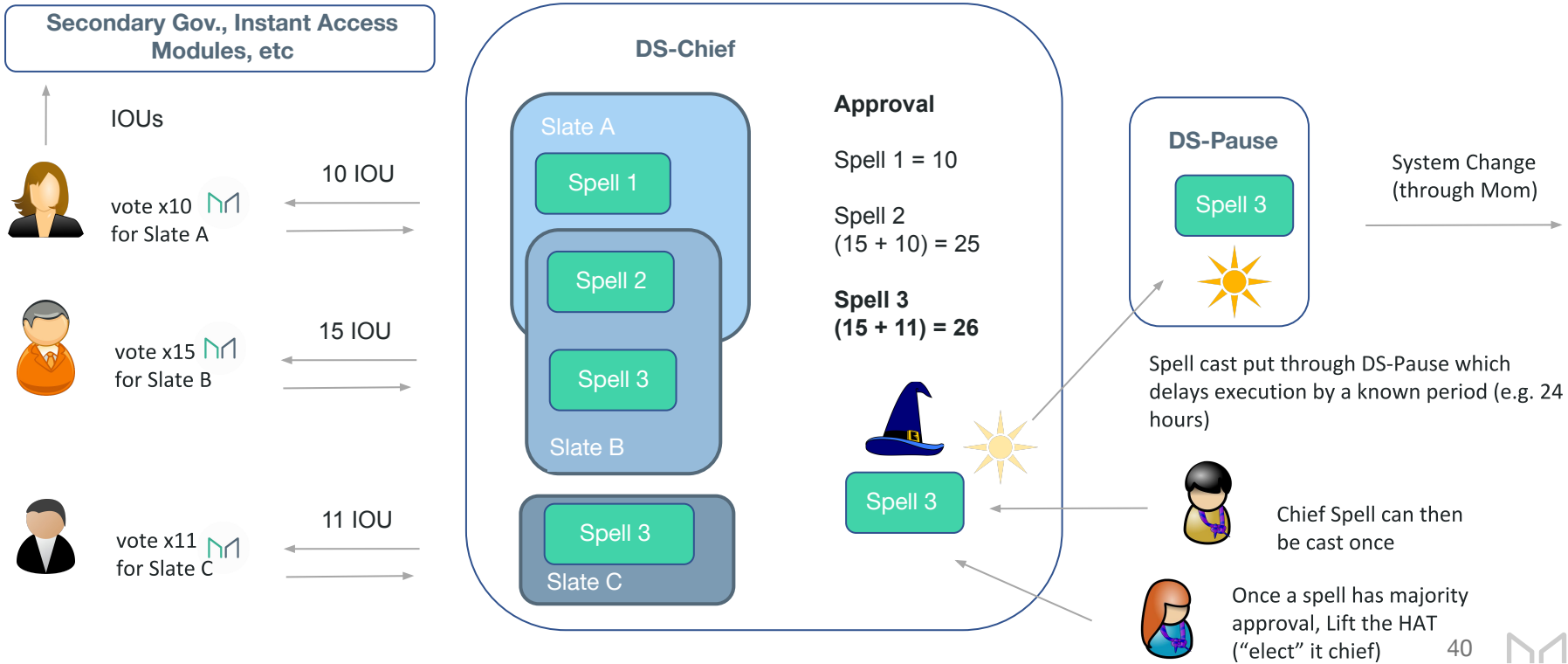
  DS-Spell is a generalized **un-owned contract that performs one action**, or series of atomic actions, one time only. The DS-SpellBook is a DS-Spell factory contract used to create Executive Spells (proposals), which perform single, preconstructed changes to the Maker Protocol. Spells can be “cast” by anyone and if the transaction is successful (no reverts or exceptional conditions), then the **spell is marked as** `done` **and cannot be re-cast**.

- **DS-Pause** - def. of Pause - “interrupt action or speech briefly.”

  Similar to the OSM, DS-Pause **schedules function calls** that can only be executed after some predetermined delay has passed. A DS-Pause implementation for Maker Governance is used to schedule Spells after their approval in DS-Chief; e.g. all Spells are executable after 24 hours. As a security component, DS-Pause ensures that those affected by governance decisions have time to react in the case of an attack. The delay period can be adjusted through an Executive Spell and extended indefinitely for coordinated upgrades to DS-Chief.

https://github.com/dapphub/ds-spell
https://github.com/dapphub/ds-pause

Governance Module - what



Rates Module - what

Compound stability fees through a rate accumulation function.

https://github.com/makerdao/dss/blob/master/src/jug.sol
https://github.com/makerdao/dss/blob/master/src/pot.sol

The Rates module is responsible for collecting stability fees on outstanding Vault debt and distributing dai earned proceeds to Savings Dai.

Rates
Jug
Pot

Components

- **Jug** - Contains `drip()`, a **public function** used to update an Ilk debt rate for an assigned stability fee. Since a debt rate is a function of time, it should be updated via `drip()` on a regular basis. Auction keepers, MKR holders, and other relevant stakeholders are **incentivized to call** `Jug.drip()` consistently.
- **Pot** - The Pot contract is where a Dai holder would lock up Internal Dai to accrue earned dai at the Dai Savings Rate. Similar to Jug, this contract employs its own Drip function used to update it's own internal rate. This rate follows the Dai Savings Rate and is used in the exchange of Savings Dai and Internal Dai. Dai holders, MKR holders, and other relevant stakeholders are **incentivized to call** `Pot.drip()` consistently.

  A portion of the Stability Fee dividends is allocated for the Dai Savings Rate by increasing the amount of Sin in the Vow at every `Pot.drip()` call. This Sin cancels out with Dai that would otherwise be sold off in surplus auctions, the mechanism for collecting stability fees.

Rates Module - why

Components

- **Jug** - Jug updates each Ilk's (collateral type) debt unit rate while the offsetting Dai is supplied to/by the Vow. The effect of this is to **apply accumulated positive/negative stability fees** to the outstanding Dai position of all Vaults. Since a blockchain is inherently passive, it requires an external "reminder" to collect stability fees; this "reminder" is in the form of calling Drip.

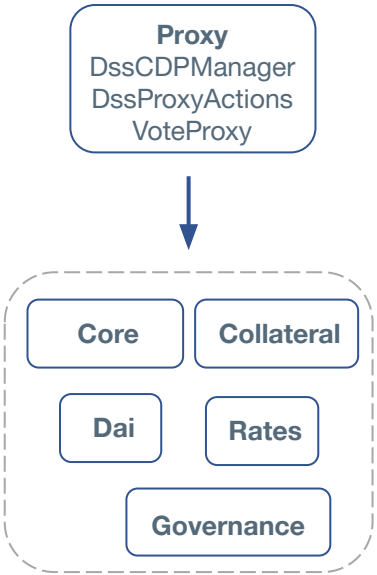
- **Pot** - Pot houses the implementation of Savings Dai, a method of **applying the dai savings rate on locked Dai.** This rate is called the Dai Savings Rate, set by Maker Governance, and is typically less than the base stability fee to remain sustainable. This purpose of Pot is to offer another incentive in holding Dai.

The 'Why' of Rates Modules is expanded in *Section 4 - Advanced Topics*

https://github.com/makerdao/dss/blob/master/src/jug.sol
https://github.com/makerdao/dss/blob/master/src/pot.sol

Proxy Module - what and why

The Proxy module is intended to increase the usability and convenience of the Maker Protocol. It contains contract interfaces, proxies, and aliases to functions necessary for Vault management and Maker governance.



Components

- **DssCdpManager** - DssCdpManager was designed to formalize the process of **Vault transfers**, enabling Vaults to be treated more like assets that can be exchanged as non-fungible tokens (NFT). It is recommended that all Vault interaction is funneled through the CdpManager. Once unlocked collateral is deposited into the Maker Protocol, the following features are available:
    - Multi Vault ownership and numerical identification (user can own *n* number of Vaults)
    - Flexible Vault transferability

https://github.com/makerdao/dss-cdp-manager

Proxy Module - what and why

- **DssProxyActions** - **A generalized wrapper for the Maker Protocol**. Utilized by the Oasis Borrow portal, DssProxyActions is a contract with functions that are to be used via a personal ds-proxy. It is similar to the Sai-Proxy and offers functions that execute a sequence of actions atomically, such as `openLockGemAndDraw(...)`.
- **VoteProxy** - The VoteProxy is a contract that **facilitates online voting with offline MKR storage**. Through a personal VoteProxy, a linked hot wallet can pull and push MKR from the proxy's corresponding cold wallet and to DS-Chief, where voting can take place with the online hot wallet. The reason for having the voting proxy contract is two-fold: to support two different voting mechanisms and to minimize the time that MKR owners need to have their wallet online.

https://github.com/makerdao/vote-proxy
https://github.com/makerdao/dss-proxy-actions

Helper - what and why

The Helper module contains smart contract helpers that handle generic patterns, such as authorization and event logging, that are inherent to the Maker Protocol and Ethereum Blockchain, respectively.

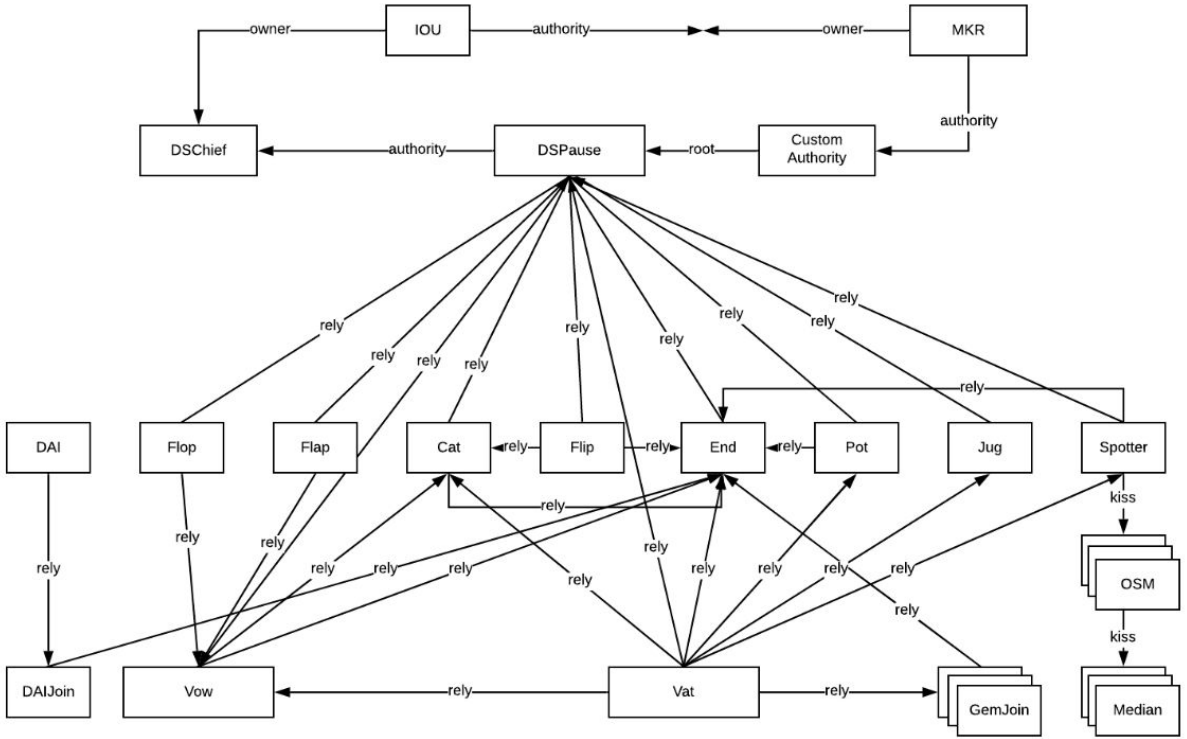
Components

Helper
DS-Auth
DS-Note

- **DS-Auth** - **Fully updatable unobtrusive auth pattern**. By design, the Maker Protocol has adopted a multi-owner authentication system, which prevents unauthorized calls that would otherwise destabilize normal operation or expose a vulnerability. Its implementation is in the form of DS-Auth, which provides a flexible and updatable `auth` modifier that restricts function-call access to the contract, contract owner, or address with granted permission via a specified `authority`.
- **DS-Note** - **Log function calls as events.** Similar to DS-Auth, DS-Note provides unobtrusive, generic function call logging by way of a `note` modifier, which triggers the capture of data as a LogNote event. Functions with this modifier will log information whenever they are called with the indexed fields being queryable by blockchain clients.

https://github.com/dapphub/ds-auth
https://github.com/dapphub/ds-note

Helper - ex of authorization within Maker Protocol



https://github.com/makerdao/dss/wiki/Auth

Example from 11/4/2019

Emergency Shutdown - what

https://github.com/makerdao/dss/blob/master/src/end.sol
https://github.com/makerdao/esm

Emergency Shutdown is the last resort to protect the system against a serious threat, such as governance attacks long-term market irrationality, hacks, and security breaches.

The Emergency Shutdown (ES) module is responsible for coordinating emergency shutdown, the process used to gracefully shutdown the Maker Protocol and properly allocate its collateral to both Vault/Dai users.

Emergency Shutdown
End
ESM

Components

- **End** - End is a smart contract that **facilitates Emergency Shutdown** within the Maker protocol; it has calling authority to much of the system and is the User's interface for claiming collateral. The ESM is authorized to initiate ES by calling the protected `cage()` function, which initiates Emergency Shutdown. It supports various scenarios, ranging from over-collateralization among all ilks to global under-collateralization, which is when the net value of all collateral types is less than the total Dai supply. In the latter edge cases, where the collateral base is limited, the Vault owner payout (in the form of excess collateral) is prioritized over the claims of Dai holders.
- **ESM** - The Emergency Shutdown Module (ESM) is **a contract with the ability to call** `End.cage()` and initiate ES. MKR holders join their funds, which are then immediately burnt. When the ESM's internal sum balance is equal to or greater than the minimum threshold, then `End.cage()` can be called.

Emergency Shutdown - why and how

Components

- **End** - Emergency shutdown is an involved, deterministic process, **requiring interaction** from all user types: Vault owners, Dai holders, Keepers, MKR governors, and other Maker Protocol Stakeholders. The high-level steps are as follows:

  1. The ESM **calls `cage()` function**, which freezes the protocol and locks spot prices (collateral USD prices) for each ilk (collateral type).

  2. Next, Vault holders interact with End to settle their Vault and **withdraw excess collateral**.

  3. After collateral auctions have concluded or been canceled and the system has settled all large Vaults, Dai holders can begin to **claim a proportional amount** of each collateral type at a fixed rate that corresponds to Dai circulation and USD value of the asset at the time ES was initiated.

  The 'Why and How' of End is expanded on in *Section 4 - Advanced Topics,* starting on **slide 63**

https://github.com/makerdao/dss/blob/master/src/end.sol

Emergency Shutdown - why and how

Components

- **ESM** - The Emergency Shutdown Module (ESM)

  Similar to DS-Chief, the ESM is a contract that locks MKR and has the ability to call `End.cage()` and initiate ES.

  It is meant to be **used by an MKR minority** to thwart two types of attack:

  1. Malicious governance
  2. An attack facilitated by a critical bug

  MKR holders **lock their MKR** in the ESM, which are then **immediately burnt**. When the ESM's internal sum balance is equal to or greater than the minimum threshold, then `End.cage()` can be called.

  If attack #1 is attempted, pledgers will have no expectation of recovering their MKR from the existing deployment (as there potentially can be a malicious majority that block the required vote). In such a case, the option is to set up an alternative fork in which the funds in which the attackers get no new MKR. If attack #2 is attempted, governance can choose to refund the ESM pledgers by minting new MKR.

https://github.com/makerdao/esm
Guide to lock MKR: https://bit.ly/2Qt8hyy

Instant Access Module - what and why

Not in scope for Nov. 18th launch of Multi-collateral Dai.
SUBJECT TO CHANGE

Instant Access Module houses the components to create direct, bounded changes to the Maker Protocol without consensus in DS-Chief.

*Names are placeholders, though concepts remain

Components

- **RO (Rates Oracle)** - Through admin access to the Rates Module, the Rates Oracle enables more dynamic updates to the Risk Premium Rates, Base Rate and Savings Rate. As a secondary governance mechanism, the Rates Oracle allows MKR holders to vote with the IOUs they receive when locking up their MKR in DS-Chief. Furthermore, to vote they have to stake some amount of Dai (e.g. $1000 or $10,000), which will be passed on to the Buffer if they vote for a losing proposal.
- **VO (NFT/LEIN Vault Oracle)** - Controlled by authorized Risk Team(s), the VO holds admin access to add an NFT/LIEN Vault type on the fly under certain restrictions.

Authorized Interface Module - what

Not in scope for Nov. 18th launch of Multi-collateral Dai.
SUBJECT TO CHANGE

The Authorized Interface Module holds the interfaces between the system and the governance contracts.

Authorized Interface*
Mom
INT-RO
INT-VO

Components

- **Mom** - Mom is a contract interface to adjust the risk parameters of the Maker Protocol. The chief in DS-Chief has the exclusive authority to call functions through Mom. The following contracts rely on Mom: Spotter, Cat, Vow, Vat, and Jug.
- **INT-RO (Rates Oracle Interface)** - Interface contract for the Rates Oracle. Accessible by the Rates Oracle. It has bounded authority over the Rates Module.
- **INT-VO (NFT/Lein Vault Oracle Interface)** - Interface contract for the NFT/LEIN Vault Oracle. It is authorized to add NFT/LEIN Vault types to the system.

*Names are placeholders, though concepts remain

Section 4

Table of Contents

- System Design Rationale
- Asset States within the Vat
- Vault States within the Vat
- Problem and solution of Compounding Fees
- Emergency Shutdown

System Design Rationale

Design Considerations

- Token Agnostic
    - System is indifferent to implementation of external tokens
    - The Join **adapters abstract away the differences** between ERC 20s, Non Fungible Tokens (NFTs), invoice tokens, etc
- Verifiable
    - Designed from bottom up to be **well suited for formal verification**; every Vat state defined and proved
    - The Vat makes ***no* external calls**, as functions in external contracts are subject to change
    - The Vat contains ***no* precision loss**; it only adds, subtracts, and multiplies
- Modular and Upgradable
    - Implementations of e.g. auctions, liquidation, Vault risk conditions, and new collateral types, to be altered on a live system through Maker Governance

Asset States within the Vat - what

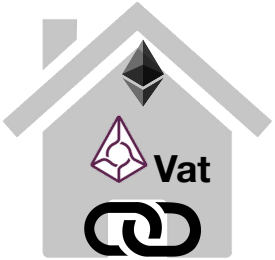
Collateral

Tokens

Unencumbered
“Unlocked”

Gem credit within the Vat. A user can increase and decrease their allocation (on a per collateral type basis) without restriction

Encumbered
“Locked”



A user has the freedom of choosing how much Gem they wish to expose to liquidation in their Vault. To do this, they lock their Gem credit in a Vault and draw Dai against their now encumbered collateral called **Ink.**

Internal Dai

Dai credit within the Vat. A user can increase and decrease their allocation through the minting and burning of Dai **without restriction**

Asset States within the Vat - why

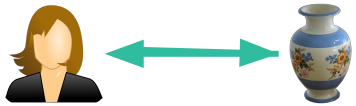
The reason why we have a separate unlocked collateral balance is for security.

To minimize dependencies, the system makes only two types of function calls to external smart contracts:

- TransferFrom( ) to deposit collateral
- Transfer( ) to free collateral

Once some collateral is deposited and registered as a gem balance, the system does not need to make any additional external calls to determine the token balance. **The user's internal unlocked balance is all they have as far as the system is concerned.**

Vault States within the Vat



Note on Step 2
Although Alice freed 7 eth from her unlocked balance, the Vault's collateralization ratio is the same in step 2 and 3, since the **Vault's debt is backed only by the amount of locked collateral.**

$$CR = \frac{Locked\ Collateral\ in\ USD}{Drawn\ Dai\ in\ USD}$$

Step	1	2	3	4
User Action	Deposit 12 Eth	Lock 5 Eth Draw 200 Dai	Free 7 eth	Wipe 200* Dai Unlock 5 Eth
Function called (Contract.method)	GemJoin.Join( )	Vat.Frob( )	GemJoin.Exit( )	Vat.Frob( )
Unlocked Collateral (Gem)	12 eth	7 eth	0	5 eth
Locked Collateral (Ink)	0	5 eth	5 eth	0
Collateral Value (USD) (Assume $100 ETH/USD)	0	$500	$500	0
Debt (Dai)	0	200	200	0
Collateralization Ratio (aka CR)	N/A	250%	250%	N/A

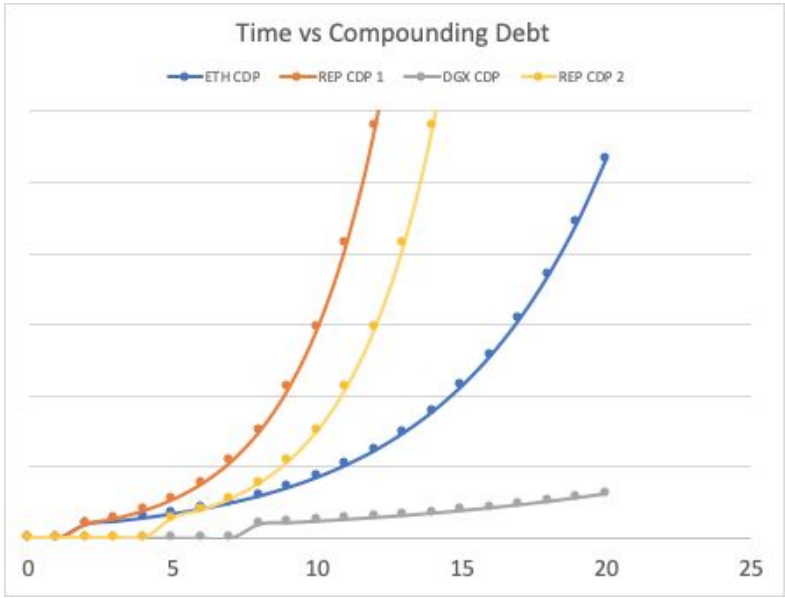
* 200 + stability fees of Dai wiped

Problem with Compounding Fees

The Maker Protocol needs to be able to calculate the total stability fee owed for each Vault, which depends on:

- How much Dai was borrowed
- How long the Dai has been outstanding
- The stability rate(s) while the Dai was outstanding (which can change at any time, even in the middle of a Vault's lifetime)

This needs to be done for all Vaults, and there could be millions of Vaults.



https://github.com/makerdao/dss/wiki/Drip

Thanks to Tyler Sorensen

Solution: Rate Accumulation Function

Keeping track of compounded stability fees for all Vaults only requires storing two types of abstract variables:

- 1 variable per Vault, **(urn.art)**, *adjusted by the user*
- 1 global variable for each collateral type, **(ilk.rate)**, *updated by calling Drip*

Calculating the total debt owed (drawn dai + stability fee accrued) for a given Vault only requires multiplying these two variables together, regardless of changes in stability rates.

The Total Debt is not stored anywhere in the contracts. Instead, when it is needed, the below calculation is performed with the urn's unique **art** and the **rate** for an individual collateral type. A common rate, used among urns of the same collateral type, sidesteps the problem of looping across all urns to update their Total Debt.

$$Total\ Debt\ (Dai) = urn.art\ x\ Ilk.rate$$

https://github.com/makerdao/dss/wiki/Drip

Thanks to Tyler Sorensen

Solution: Rate Accumulation Function

$$Total\ Debt\ (Dai) = urn.art\ x\ Ilk.rate$$

urn.art

What - Abstract variable for an urn. It is equivalent to the urn's total debt when rate equals 1.

How is it updated? - It is adjusted when Debt is drawn/wiped by the urn owner.

$$art = old\ art \pm \frac{additional\ Debt\ drawn/wiped}{rate}$$

Notes - The user must have unlocked/locked collateral (urn.ink) in the Vat to increment/decrement art

ilk.rate

What - Abstract variable for an Ilk. When a new collateral type is added to the Vat, it is set to 1. It is a function of stability rate and time.

How is it updated? - Every time Drip is called.

$$rate = old\ rate * (1 + Stability\ Rate)^{(Time\ since\ the\ last\ Drip\ call)}$$

Notes - As long as the stability fee is positive, Ilk.rate will *increase indefinitely.*

Concept: Rate Accumulation Function

Case: Ilk.rate is updated

An Ilk.rate vs. time graph follows a generalized rate curve, compounded on a per second basis. Every Drip call updates the actual rate and brings it back to this ideal rate curve.

Due to gas costs and the tragedy of the commons problem with the public Drip function, the update frequency could be recurring but irregular.

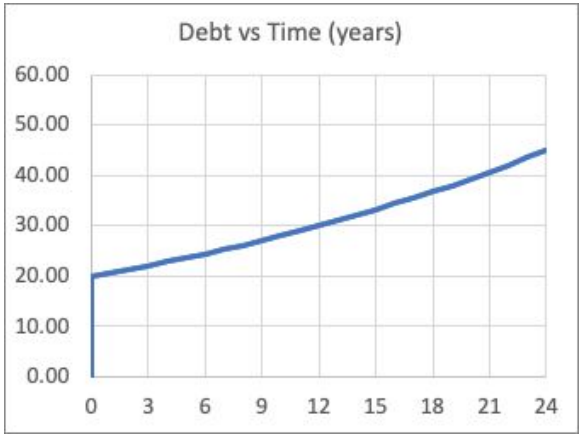
As the system matures, more stakeholders with more stake will ensure that Drip is called more and the ilk.rate(s) follow the ideal rate curve(s) more closely.

As a consequence, a Vault opened and closed between drip calls (i.e. rate updates) would avoid stability fees.

[Compound Stability Fee Example]

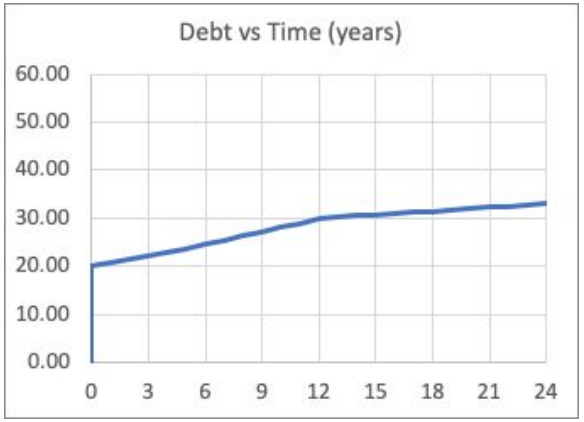
Concept: Rate Accumulation Function

Case: Stability Rate is updated



Step 1: 50% APR
Draw 20 Dai at t = 0

Step 2: 10% APR
Rate Decrease to 10% at t = 12

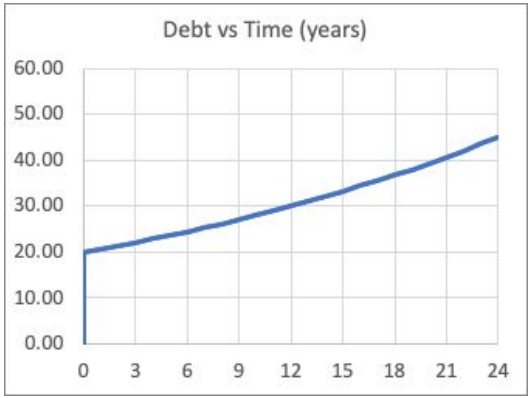


When the stability fee is updated, the next ilk.rate update (i.e. Drip call) would simply use the new 'Stability Rate'. This varies the ilk.rate curve, which has an effect on all Vault debt curves. As long as the stability rate is positive, Ilk.rate will increase indefinitely. Again, in favor of simplicity and storage, the system does not record the stability rate history for each Ilk.

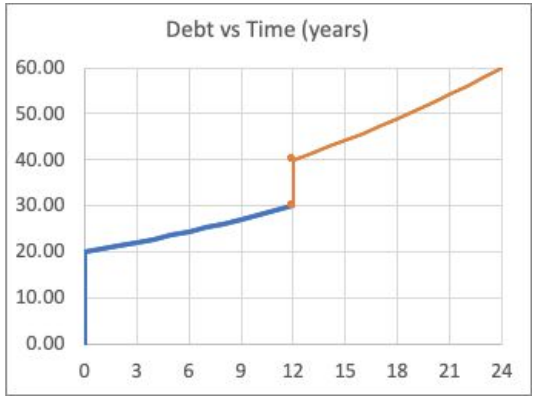
$$rate = old\ rate * (1 + Stability\ Rate)^{(Time\ since\ the\ last\ Drip\ call)}$$

Concept: Rate Accumulation Function

Case: Additional Dai is drawn before debt is wiped



Step 1: 50% APR
Draw 20 Dai at t = 0
Note: at t =0, rate is 1



Step 2: 50% APR
Draw 20 Dai at t = 0
Draw 10 Dai at t = 12

The Vat only records total debt when rate is 1, which is simply urn.art

When additional Dai (debt) is drawn against a Vault, the Debt curve shifts vertically by the amount of additional dai drawn. However, the Vat can only record a change in **urn.art**. Thus, the amount of additional **urn.art** is equivalent to the amount of additional Dai drawn, valued back to when **Ilk.rate = 1**. (Similar to the Time Value of Money concept). The **total debt** (at t=0 and rate=1) that corresponds to the adjusted debt profile in Step 2 is equivalent to 26.667 Dai (green line). Thus, in favor of simplicity and storage, the system does not record the Vault's draw/wipe history.

Emergency Shutdown - Design Characteristics

- **Dai no-race condition** - every Dai holder will be able to redeem the same quantity of collateral, regardless of when they interact with the contract.
- **Vault Parity** - Vault Owners are prioritized and are allowed to redeem their excess collateral before Dai holders.
  - At the time of Emergency Shutdown (ES), individual Vaults, entire collateral types, or the Maker protocol can be undercollateralized, which is when the value of debt exceeds the value of collateral ("negative equity")
  - Maker's current implementation favors Vaults owners in all cases by allowing them to free their entire amount of excess collateral. Thus, in the low likelihood event that any Vaults become undercollateralized, the Dai holders receive a "haircut" to their claim on collateral. In other words, Dai holders' claim may be less than a dollar's worth of collateral.
- **Immediate Vault redemption** - After ES is initiated, Vault owners are allowed to free their collateral immediately, provided that they execute all contract calls atomically.
- **No off-chain calculations** - The system does not require the cage authority to supply any off-chain calculated values (e.g. it can rely entirely on the last OSM feed prices).
- **Vow Buffer Assistance** - After ES is initiated, any surplus (and bad debt) in the buffer acts as a reward (and penalty) distributed pro-rata to all Dai Holders. e.g. if 10% of total system debt is in the form of net surplus in the Vow, then Dai holders receive 10% more collateral.

Emergency Shutdown - User Stories

Regardless of Vault delegation (`wish(owner, manager)`), Vault redemption will be performed by the owners. Most Dai holders are assumed to sell to secondary markets. Dai redemption will likely be completed by Redemption (or Super) Keepers that are better equipped to:

- Find liquidity for all collateral types
- Handle permissioned collateral that require KYC, such as tokenized securities

How will each agent type interact with the End Contract after Emergency Shutdown has been initiated?

- **Vault Owners of Single Ilk type**
- **Dai Holders**
- **Maker Protocol Stakeholders**

Preface to the following stories:

- All stories start with Step #0 `cage( )`, which is called by the Emergency Shutdown Module (ESM)
- Each contract call assumes that it has not been called already
- Undercollateralized Vaults = debt > collateral or `urn.art * ilk.rate > urn.ink * (ilk.spot * ilk.mat)`

This slide requires familiarity of Emergency Shutdown (ES). Due to its complexity, the design and operation of ES is [elaborated here]

Emergency Shutdown - User Stories cont'd

Vault Owners of single ilk type

I own a Vault and wish to free my excess collateral from the Maker Protocol.

Overcollateralized Vaults

1. Value the collateral in their Vault → `cage(ilk)`
2. Settle their outstanding debt with their locked collateral → `skim(ilk, urn)`
3. Unlock excess collateral in Vault → `free(ilk)`

Overcollateralized Vaults that were bitten and are in the "Tend" phase of the respective Collateral auction

1. Value the collateral in their Vault → `cage(ilk)`
2. Either wait until their tend auction phase finishes OR speed up collateral processing → `skip(ilk, id)`
3. Settle their outstanding debt with their locked collateral → `skim(ilk, urn)`
4. Unlock excess collateral in Vault → `free(ilk)`

Undercollateralized Vaults → not incentivized to do anything

This slide requires familiarity of Emergency Shutdown (ES). Due to its complexity, the design and operation of ES is elaborated here

Emergency Shutdown - User Stories cont'd

Dai holders

I want to convert my Dai into its claim on the underlying collateral basket.

1. Transition system into the Dai withdrawal phase → `thaw()`
2. Set the Dai-to-collateral exchange rate (`fix[ilk]`) for each collateral type → `flow(ilk)`
3. Send all their Dai to the Maker Protocol → `pack(wad)`
4. Retrieve a portion of each collateral type → `cash(ilk, wad)`

Maker Protocol Stakeholders (Larger Dai holders/custodians, MKR holders, Redemption keepers, etc.)

I want to ensure that emergency shutdown is completed and accounts for all Vaults. Cage-Keeper
Note that their story may overlap with the above stories.

1. Yank all flop and flap auctions → `Flop.yank(id)` and `Flap.yank(id)` for multiple flop and flap auctions
2. Value the collateral in all Vault types → `cage(ilk)` for multiple collateral types
3. Either wait until their tend auction phase OR speed up collateral processing → `skip(ilk, id)` for multiple flip auctions over all collateral types
4. Settle all over/under collateralized Vaults → `skim(ilk, urn)` for multiple Vaults
5. Transition system into the Dai withdrawal phase → `thaw()`
6. Set the Dai-to-collateral exchange rate (`fix[ilk]`) for each collateral type → `flow(ilk)` for multiple collateral types

This slide requires familiarity of Emergency Shutdown (ES). Due to its complexity, the design and operation of ES is elaborated here

References

Maker Foundation Team

Code, readme, and wiki - https://github.com/makerdao/dss

K specification of smart contracts - https://github.com/makerdao/k-dss

Whitepaper - https://makerdao.com/whitepaper

The End

4

Thank you for taking the time to go through this presentation!

Please reach out on **Reddit**, **RocketChat**, or the **forum.makerdao.com**, if any questions arise

Submit errors to kenton@makerdao.com or @Kenton on Rocket Chat