CDP Manager - Detailed Documentation

Managing Vaults to be transferred between users * Contract Name: * cdpManager.sol * Type/Category: * Vault Management * Associated MCD System Diagram * Contract Source * Etherscan *

1. Introduction (Summary)

Summary: TheDssCdpManager (akamanager) was created to enable a formalized process for Vaults to be transferred between owners, much like assets are transferred. It is recommended that all interactions with Vaults be done through the CDP Manager. Once unlocked collateral has been deposited into the Maker Protocol, users can make use of the following features:

- Multi Vault ownership and numerical identification (users can own N number of Vaults)
- Vault transferability
-

?

Note: The MCD system diagram above shows that the Vault user goes through the proxy in order to interact with the CDP Manager but it is also possible to directly use the CDP Manager contract.

1. Contract Details

Key Functionalities (as defined in the smart contract)

- cdpAllow(uint cdp, address usr, uint ok)
- : Allow/Disallow (ok
-) ausr
- address to manage thecdp
- .
- urnAllow(address usr, uint ok)
- : Allow/Disallow (ok
-) ausr
- address to interact with an urn for the purposes of either entering (src
-) or quitting (dst).
- open(bytes32 ilk, address usr)
- : Opens a new Vault forusr
- to be used for anilk
- collateral type.
- give(uint cdp, address dst)
- : Transferscdp
- todst
- .
- frob(uint cdp, int dink, int dart)
- : Increments/decrements theink
- amount of collateral locked and increments/decrements theart
- amount of debt in thecdp
- depositing the generated DAI or collateral freed in thecdp
- address.
- frob(uint cdp, address dst, int dink, int dart)
- : Increments/decrements theink
- amount of collateral locked and increments/decrements theart
- amount of debt in thecdp
- depositing the generated DAI or collateral freed into aspecified
- dst
- address.
- flux(bytes32 ilk, uint cdp, address dst, uint wad)
- : Moveswad
- (precision 18) amount of collateralilk
- fromcdp
- todst
- .
- flux(uint cdp, address dst, uint wad)
- : Moveswad
- amount ofcdp
- collateral fromcdp
- todst
- .

- move(uint cdp, address dst, uint rad)
- : Movesrad
- (precision 45) amount of DAI fromcdp
- todst
- .
- quit(uint cdp, address dst)
- : Moves the collateral locked and debt generated fromcdp
- todst
- .
-

Note: dst refers to the destination address.

Storage Layout

- vat
- : core contract address that holds the Vaults.
- cdpi
- : Auto incremental id.
- urns
- : MappingCDPId => UrnHandler
- list
- : MappingCDPId => Prev & Next CDPIds
- (double linked list)
- owns
- : MappingCDPId => Owner
- ilks
- : MappingCDPId => Ilk
- (collateral type)
- first
- : MappingOwner => First CDPId
- last
- : MappingOwner => Last CDPId
- count
- : MappingOwner => Amount of CDPs
- allows
- : MappingOwner => CDPId => Allowed Addr => True/False
-

1. Key Mechanisms & Concepts

Summary

The CDP Manager was created as a way to enable Vaults to be treated more like assets that can be exchanged. Originally, thedss core contracts did not have the functionality to enable transferring Vault positions. The CDP Manager was created to wrap this functionality and enable transferring between users.

High-level Purpose

- Themanager
- receives thevat
- address in its creation and acts as an interface contract between it and the users.
- Themanager
- keeps an internal registry ofid => owner
- andid => urn
- allowing for theowner
- to executevat
- functions for theirurn
- via themanager
- .
- Themanager
- keeps a double linked list structure that allows the retrieval of all the Vaults that anowner
- has via on-chain calls.
-
    - In short, this is what theGetCdps
-
    - is for. This contract is a helper contract that allows the fetching of all the Vaults in just one call.
- *
-

CDPManager Usage Example (common path):

- A User executesopen
- and gets aCDPId
- in return.
- After this, theCDPId
- gets associated with anurn
- withmanager.urns(cdpId)
- and thenjoin
- 's collateral to it.
- The user can then executefrob
- to choose whichdst
- address they want to use to send the generated DAI to.
- If the user executesfrob
- withoutdst
- then the generated DAI will remain in the Vault'surn
- . In this case, the user canmove
- it at a later point in time.
-
    - Note that this is the same process for collateral that is freed afterfrob
-
    - (for thefrob
-
    - function that doesn't require thedst
-
    - address). The user canflux
-
    - it to another address at a later time.
- *
- In the case where a user wants to abandon themanager
- , they can usequit
- as a way to migrate their position of their Vault to anotherdst
- address.
-

1. Gotchas (Potential source of user error)

2. For the developers who want to integrate with themanager

3. , they will need to understand that the Vault actions are still in theurn
4. environment. Regardless of this, themanager
5. tries to abstract theurn
6. usage by aCDPId
7. . This means that developers will need to get theurn
8. (urn = manager.urns(cdpId)
9.) to allow thejoin
10. ing of collateral to that Vault.
11. As themanager
12. assigns a specificilk
13. perCDPId
14. and doesn't allow others to use it for theirs, there is a secondflux
15. function which expects anilk
16. parameter. This function has the simple purpose of taking out collateral that was wrongly sent to a Vault that can't handle it/is incompatible.
17. Frob Function(s):
18.
    - When youfrob
19.
    - in the CDP manager, you generate new DAI in thevat
20.
    - via the CDP manager which is then deposited in theurn
21.
    - that the CDP manager manages. This process depends on whichfrob
22.
    - function you use (there existtwo
23.
    - frob
24.
    - functions). In short, one allows a destination address and the other doesn’t require it.

25.
    - If you use thefrob
26.
    - function that has the destiny (dst
27.
    -) address, you are saying that you can send any Dai generated or collateral that has been freed. The secondfrob
28.
    - function is meant for leaving the collateral in theurn
29.
    - address because theurn
30.
    - is owned by the CDP manager. In this case, you would need to manually use theflux
31.
    - ormove
32.
    - functions to get the DAI or collateral out. These functions (flux
33.
    - andmove
34.
    -) may be more beneficial for a developer working with the proxy function, as it allows for more flexibility. For example, by using these functions you can move a specific amount of collateral and can use the other functions to do it. Overall, it can make working with it a little more flexible on specific developer needs.
35. *
36. As mentioned above in the summary, thedss
37. core contracts originally did not have the functionality to enable the transfer of Vault positions. Since then, the core contracts have also implemented a native transfer functionality calledfork
38. which allows the transferring of a Vault to another address. However, there is a restriction, which is that the address owner that will be receiving the Vault needs to provide authorization that they do in fact want to receive it. This was created for the situation when a user is transferring the collateral that is locked as well as the debt generated. If you are simply moving collateral to another address, there is no issue but in the case that you are also transferring the debt generated, there is a chance of putting a perfectly safe Vault in a risky position. This makes the contract functionality a little more restrictive. Therefore, the CDP manager is a good option to keep a simple way of transferring Vaults and recognizing them via a numeric ID.
39.

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

Potential Issues around Chain Reorganization

Whenopen is executed, a newurn is created and acdpId is assigned to it for a specificowner . If the user usesjoin to add collateral to theurn immediately after the transaction is mined, there is a chance that a reorganization of the chain occurs. This would result in the user losing the ownership of thatcdpId /urn pair, therefore losing their collateral. However, this issue can only arise when avoiding the use of theproxy functions via aprofile proxy as the user willopen thecdp andjoin collateral in the same transaction.

Previous Vote Proxy - Detailed Documentation Next DSR Manager - Detailed Documentation Last updated4 years ago On this page * 1. Introduction (Summary) * 2. Contract Details * Key Functionalities (as defined in the smart contract)* Storage Layout * 3. Key Mechanisms & Concepts * Summary * High-level Purpose * CDP Manager Usage Example (common path): * 4. Gotchas (Potential source of user error) * 5. Failure Modes (Bounds on Operating Conditions & External Risk Factors) * Potential Issues around Chain Reorganization

Export as PDF