Join - Detailed Documentation

- Contract Name:
- join.sol
- Type/Category:
- DSS —> Token Adapter Module
- Associated MCD System Diagram
- Contract Source
- Etherscan
-
  - Join Dai
-
  - Join Eth
-
  - Join Bat
- *
-

1. Introduction (Summary)

Join consists of three smart contracts:GemJoin ,ETHJoin , andDaiJoin: GemJoin - allows standard ERC20 tokens to be deposited for use with the system.ETHJoin - allows native Ether to be used with the system.DaiJoin - allows users to withdraw their Dai from the system into a standard ERC20 token.

Eachjoin contract is created specifically to allow the given token type to bejoin 'ed to thevat . Because of this, eachjoin contract has slightly different logic to account for the different types of tokens within the system.

?

1. Contract Details:

Glossary (Join)

- vat
-
  - storage of theVat
- ’s address.
- ilk
-
  - id of the Ilk for which aGemJoin
- is created for.
- gem
-
  - the address of theilk
- for transferring.
- dai
-
  - the address of thedai
- token.
- one
-
  - a 10^27 uint used for math inDaiJoin
- .
- live
-
  - an access flag for thejoin
- adapter.
- dec
-
  - decimals for the Gem.
-

Everyjoin contract has 4 public functions: a constructor,join ,exit , andcage . The constructor is used on contract initialization and sets the core variables of thatjoin contract.Join andexit are both true to their names.Join provides a mechanism for users to add the given token type to thevat . It has slightly different logic in each variation, but generally resolves down to atransfer and a function call in thevat .Exit is very similar, but instead allows the the user to remove their desired token from thevat .Cage allows the adapter to be drained (allows tokens to move out but not in).

1. Key Mechanisms & Concepts

TheGemJoin contract serves a very specified and singular purpose which is relatively abstracted away from the rest of the core smart contract system. When a user desires to enter the system and interact with thedss contracts, they must use one of thejoin contracts. After they have finished with thedss contracts, they must callexit to leave the system and take out their tokens. When theGemJoin getscage d by anauth ed address, it canexit collateral from the Vat but it can no longerjoin new collateral.

User balances for collateral tokens added to the system viajoin are accounted for in theVat asGem according to collateral typeIlk until they are converted into locked collateral tokens (ink) so the user can draw Dai.

TheDaiJoin contract serves a similar purpose. It manages the exchange of Dai that is tracked in theVat and ERC-20 Dai that is tracked byDai.sol . After a user draws Dai against their collateral, they will have a balance inVat.dai . This Dai balance can beexit ' ed from the Vat using theDaiJoin contract which holds the balance ofVat.dai and mint's ERC-20 Dai. When a user wants to move their Dai back into theVat accounting system (to pay back debt, participate in auctions, packbag 's in theEnd , or utilize the DSR, etc), they must callDaiJoin.join . By callingDaiJoin.join this effectivelyburn 's the ERC-20 Dai and transfersVat.dai from theDaiJoin 's balance to the User's account in theVat . Under normal operation of the system, theDai.totalSupply should equal theVat.dai(DaiJoin) balance. When theDaiJoin contract getscage 'd by anauth 'ed address, it can move Dai back into the Vat but it can no longerexit Dai from the Vat.

1. Gotchas (Potential source of user error)

The main source of user error with theJoin contract is that Users should nevertransfer tokens directly to the contracts, theymust use thejoin functions or they will not be able to retrieve their tokens.

There are limited sources of user error in thejoin contract system due to the limited functionality of the system. Barring a contract bug, should a user calljoin by accident they could always get their tokens back through the correspondingexit call on the givenjoin contract.

The main issue to be aware of here would be a well-executed phishing attack. As the system evolves and potentially morejoin contracts are created, or more user interfaces are made, there is the potential for a user to have their funds stolen by a maliciousjoin contract which does not actually send tokens to thevat , but instead to some other contract or wallet.

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

There could potentially be a vat upgrade that would require new join contracts to be created.

If agem contract were to go through a token upgrade or have the tokens frozen while a user's collateral was in the system, there could potentially be a scenario in which the users were unable to redeem their collateral after the freeze or upgrade was finished. This scenario likely presents little risk though because the token going through this upgrade would more than likely want to work alongside the Maker community to be sure this was not an issue.

Previous Collateral Module Next Liquidation 2.0 Module Last updated4 years ago On this page *1. Introduction (Summary) * 2. Contract Details: * Glossary (Join) * 3. Key Mechanisms & Concepts * 4. Gotchas (Potential source of user error) * 5. Failure Modes (Bounds on Operating Conditions & External Risk Factors)

Export as PDF