Voting with a 2-Key Contract

The UMA DVM two-key contract enables you to actively engage in voting using hot keys (private keys stored on machine and used often), but protect your funds with the stronger, and preferable, use of cold keys (private keys stored offline and used rarely). Another way this contract can be used is to delegate voting rights to an EOA, while still maintaining token ownership with a multisig or other ownership structure.

Each two-key contract is a unique smart contract that you deploy. It keeps track of:

- The number of UMA tokens you have deposited into the two-key contract
- The address of your hot wallet; the hot wallet is permissioned to sign transactions and vote in DVM governance
- The address of your cold wallet; only the cold wallet (and importantly, not the voting wallet or any other address) is permissioned to withdraw UMA tokens from the two-key contract
-

DANGER

If you do not know exactly why you are setting up a 2-Key contract, then you likely do not need one. Deploying the contract is expensive and unnecessary unless you need to vote with separate hot and cold keys.

Voting with a hardware wallet is not a reason that necessitates needing to set up a 2-Key contract. You can likely vote by connecting your hardware wallet to MetaMask or another wallet provider.

If you have any doubt about your need for a 2-Key contract, please ask in the UMA Discord before proceeding.

Prerequisites

- You need ETH in your hot wallet for the initial deployment of the 2-key contract and for submitting all subsequent votes.
- You need ETH in your cold wallet to make the initial UMA token deposit, and also if you ever want to withdraw tokens from the 2-key contract.
-

Instructions to deploy your own 2-key contract

Connect

Navigate to the UMA voter dApp (vote.umaproject.org) and connect your MetaMask hot wallet. This will be your voting wallet going forward.

Deploy

Click on the gear icon on the far right.

In the modal, clickAdd Cold Wallet Address

Input your cold wallet address and clickSave .

Verify your deployment was correct

The following steps help you verify that your two-key contract was deployed correctly and check that the permissions are held by the correct address.

- Copy the 2-key smart contract address (i.e., DesignatedVoting Escrow Account) from the voter dApp and view it on Etherscan
- UnderRead Contract
- input roleID0
- in function1. getMember
- , and click query. Check that the address output matches your cold wallet address.
- Input roleID1
- in function1. getMember
- , and click query. Check that the address output matches your hot wallet address.
-

Voting

To vote with the 2-key contract, you will need to send UMA tokens to it. After sending UMA tokens to the two-key contract address, you can immediately start voting with your hot keys at the voter dApp. This will exclude you from voting with any additional UMA tokens that are in your hot wallet through the voter dApp.

Withdraw tokens

The following steps let you withdraw UMA tokens from the two-key contract using your cold keys. You will have to withdraw and deploy a new two-key contract if you want to designate a new hot key for voting.

- Navigate to your DesignatedVotingContract on Etherscan
- Connect your cold wallet by clicking the “Connect to web3” button
- Use function11. withdrawErc20
- Input erc20Address:0x04fa0d235c4abf4bcf4787af4cf447de572ef828
- Input the number of tokens that you want to withdraw * 10^18 (for example, if you want to withdraw 1.1 tokens, you should input1100000000000000000
-)
- Click “Write”
- Confirm the transaction with your cold keys and wait for the transaction to finish mining
-

Changing the voter address

To change the voter address for a DesignatedVotingContract, you just need to callresetMember(newAddress) with your cold keys (owner address) and pass in the new address that you wish to use as the voter.

Previous Event-Based Prediction Market Next Unsupported Contracts Last updated1 month ago On this page * Prerequisites * Instructions to deploy your own 2-key contract

Was this helpful? Edit on GitHub