# Review

* Review for consistent terms (revision, proposed addition/removal, need any others?)
* Any holes? Add to overview? Add short intro to each How to section.

# 

# Overview

decentraList is a platform for creating customizable, decentrally decided, updating on-chain Lists of Ethereum addresses for reference in other smart contracts.

**Properties:**

**Customizable -** You write the criteria for an address to be on the list. You can also set and fund rewards for anyone to add or remove addresses from the list to incentivize.

**Decentrally Decided -** All proposed additions and removals to the list are verified as per the list criteria by [UMA’s Optimistic Oracle](https://docs.umaproject.org/), not a centralized, trusted party.

**Updating -** The List is referenceable by other contracts upon creation and can be revised as per the List criteria into the future while staying referenceable. There are optional rewards that be set for adding and/or removing addresses from the list, incentivizing anyone to keep the list current.

**On-chain -** the List of addresses is stored on a smart contract as a mapping for easy, low gas referencing by other smart contracts.

# How to:

## Create a New List

1. Select your desired Ethereum network in the top right corner and connect your wallet
2. Hit the Create List Button in the top right corner
3. Fill in the following fields:
   * Title: a short descriptive title for the List. ie. XYZ Hackers
   * List Criteria: detailed criteria describing what addresses should be included in the list. The List Criteria will be passed to the UMA Optimistic Oracle whenever addresses are proposed to be added or removed from the list.   
     Note: The criteria must be publicly verifiable so Oracle participants can verify proposed revisions.
   * Token Address: the smart contract address of the token that will be used for Oracle bonds and any optional List rewards. All tokens on the UMA Optimistic Oracle whitelist for the chosen network are accepted.   
     Note See UMA token whitelists here: [Mainnet](https://docs.umaproject.org/resources/approved-collateral-types) | [Goerli](https://goerli.etherscan.io/address/0x63fDfF29EBBcf1a958032d1E64F7627c3C98A059#readContract#F1)
   * Owner Address: the address of the List owner. Owner’s have the ability to revise the Bond, Addition Reward, Removal Reward and Liveness period, as well as remove funds in the List contract. The Owner has no privileges to add or remove addresses from the List.

Note: If it is desired to make the list a public good where multiple parties contribute funds to the contract for paying rewards and no party should be able to remove funds, ownership can be renounced anytime after creating the list.

* + Bond: When addresses are proposed to be added or removed from the list, the proposer must deposit a Total Bond to the Oracle. The Total Bond is UMA’s fixed Final Fee plus this Bond. The Bond amount must be greater or equal than the Final Fee.   
    Note: The Total Bond is lost if the Oracle finds that the proposed addition or removal of addresses was incorrect, but fully refundable otherwise.   
    Note: Setting a higher Bond increases security by increasing the incentives Oracle disputer’s have to find incorrect proposals. However, higher bonds also increase the risk and capital requirements for List revision proposers.
  + Addition Reward (optional): the amount paid from the List contract to proposer per address (not transaction) successfully added the list.   
    Note: The number of addresses added to the list is calculated when an Oracle verified proposal is executed on the List smart contract.   
    Note: Paying a higher Addition Reward incentivizes anyone to propose additions ot the list.
  + Removal Reward (optional): the amount paid from the List contract to proposer per address (not transaction) successfully removed from the list.   
    Note: The number of addresses removed from the list is calculated when an Oracle verified proposal is executed on the List smart contract.   
    Note: Paying a higher Addition Reward incentivizes anyone to propose additions ot the list.
  + Liveness Period: the time the proposed changes to the list will stay open on the Oracle for disputes. The minimum Liveness Period varies by network (1 sec for Goerli, 8 hours for Mainnet).   
    Note:Longer Liveness Period’s gives better assurance that the proposals are thoroughly verified, but also decreases the response time for proposed changes to the List.

1. Click Submit and sign the transaction in your wallet.

## Add or Remove Addresses

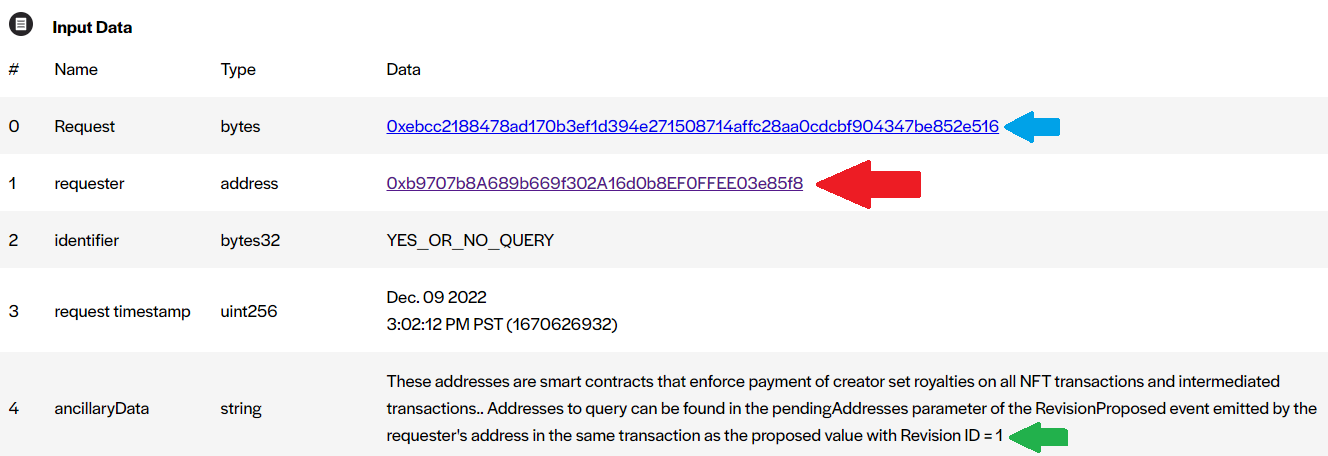
1. Select your desired Ethereum network in the top right corner and the List to revise in the dropdown.
2. Click the Add Addresses or Remove Addresses button.
3. Approve the List contract to spend the Total Bond amount by clicking the Submit Approval Button and signing the transaction in your wallet.
4. Enter the addresses to Add or Remove. Up to 99 addresses can be included in one proposal.
5. Click submit Transaction and sign the transaction in your wallet.
   * Note: the Oracle will review the proposed addition or removal as a whole, so if the proposal includes one address that is incorrect, the entire proposal will be rejected, and the proposer’s Total Bond lost.
   * Note: the Addition and Removal Rewards are calculated based on the number of addresses revised on the List and paid at the time of execution. If the List contract does not have sufficient balance to pay Rewards due, the full proposal is still executed and the proposer will receive the List’s balance.

## Verify Oracle Requests

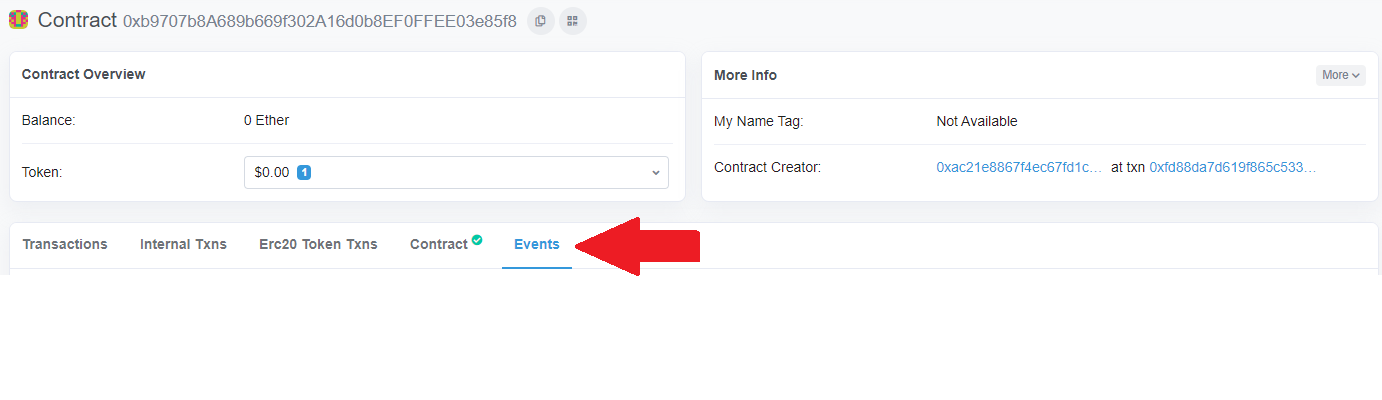
UMA Optimistic Oracle price requests made by decentraList are handled by the Oracle the same as any other price requests. For more info on how the Oracle works, see UMA’s [docs](https://docs.umaproject.org/).

The ancillary data for decentraList price requests will start with the List Criteria specific to the List. The addresses to review are not included in the ancillary data as up to 99 addresses may be proposed at once. After the specific List Criteria, there are general instructions to find the addresses included in the price request. Below are suggested steps to expand on the general instructions:

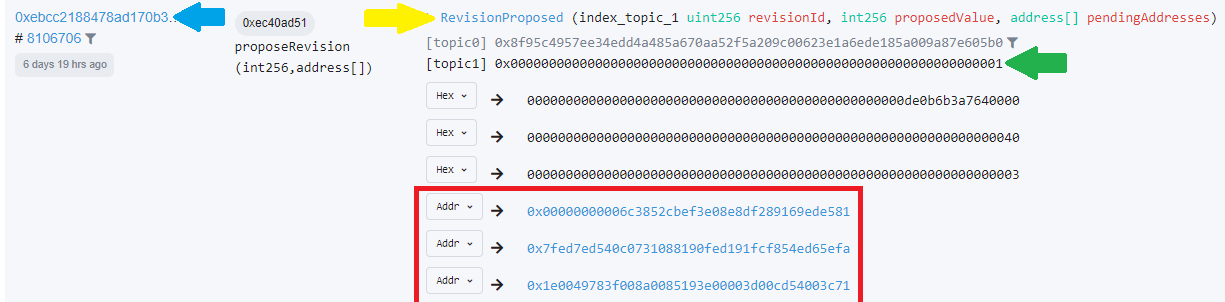
1. Click on the requester address (red arrow below) to open up the List smart contract in Etherscan.



1. Click on the Events tab (red arrow below) on the Etherscan



1. Find the RevisionProposed event (yellow arrow below) with:
   * Revision ID (green arrow below) that matches the Oracle ancillary data (green arrow above)  
     AND
   * transaction hash (blue arrow below) that matches the Oracle request field (blue arrow above).

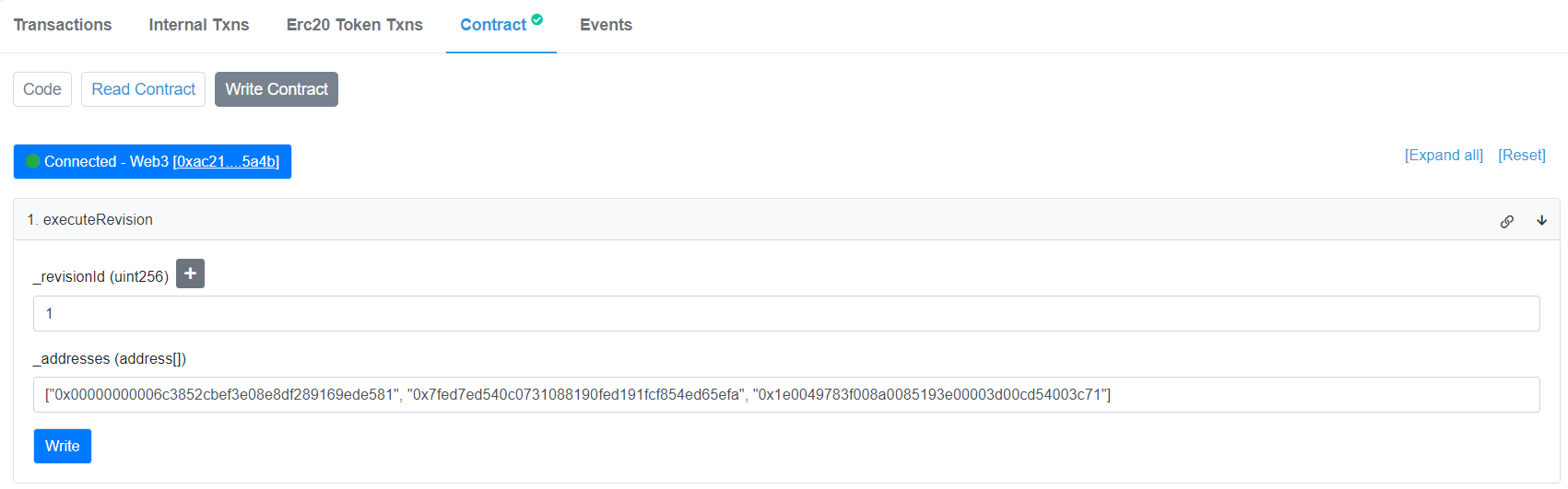


1. The addresses to review will be in the red box shown below. You can set the data to show in address format by setting the drop downs on the left side of the red box.

## Execute Oracle Approved Revisions

Once the Oracle settles a price on a proposed revision, the revision will be marked on the List smart contract as Approved. The executeRevision function must be called on the List contract to pay out any rewards and add the addresses to the List. Any address can call this function. Currently this must be manually called, but in the future decentraList plans to run a bot listening for approved revisions that immediately executes them.

To manually execute the approved revision follow these steps:

1. Find the Revision ID for the ApproveRevision event that you would like to execute.
2. Get the addresses proposed in the ProposeRevision event with the same Revision ID from #1 above. See Verify Oracle Requests #3 above for more info on finding the proposed addresses.
3. Call the executeRevision function passing in the Revision ID and an array of addresses from the ProposeRevision event. The array of addresses must be exactly the same and in the same order as listed in the ProposeRevision event.   
     
   Below is a screenshot example of calling executeRevision from Etherscan:  
     
   

Note: the Revision ID and address array match the example from the Verify Oracle Requests section above.

## Reference a List in another Smart Contract

1. Download the Decentralist Interface [here](https://github.com/pumpedlunch/decentraList/blob/master/contracts/interfaces/DecentralistInterface.sol) and save to your project repo.
2. Import the interface into your smart contract
3. Declare a Decentralist variable set to the specific List smart contract address wrapped in the Decentralist ABI
4. Call the noList() function on the Decentralist variable with the address to lookup as the only argument

See example contract which references a List:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.2;

import "./DecentralistInterface.sol"; //import DecentralistInterface

contract WhiteListVote {

DecentralistInterface public decentralist; //declare decentralist variable of type DecnetralistInterface

uint256 public yesCount;

uint256 public noCount;

mapping(address => bool) public hasVoted;

constructor(address list) {

decentralist = DecentralistInterface(list); //set address for decentralist

}

function vote(bool \_vote, address \_address) external {

require(decentralist.onList(\_address), "msg.sender must be on current voting whitelist");

//call decentralist.onList to return bool whether \_address is on list

require(hasVoted[\_address] == false, "msg.sender has already voted");

hasVoted[\_address] = true;

if(\_vote) {

yesCount++;

} else {

noCount++;

}

}

}

## Verify Full Address List

The addresses that are currently on a given List are stored on-chain in a mapping rather than an array to save on transaction gas costs. The full List of addresses for a given List can be viewed on the decentraList dapp. The full List of addresses can be verified using the following methodology:

1. Querying all RevisionExecuted events for one List smart contract
2. The events must be looped over in chronological order adding or removing all revisedAddresses from the event depending on whether the proposed value is 1E18 or 0 respectively. The zero address should be ignored in revisedAddresses and addresses already on the list should not be added in duplicate.

# Resources

Github Repo:

<https://github.com/pumpedlunch/decentraList>

Deployed Addresses: <https://github.com/pumpedlunch/decentraList/blob/master/public/deployedAddresses.txt>

Contact:

<https://twitter.com/pumpedlunch>