**Overview**

This Solidity smart contract implements a basic voting system for executing proposals on the Ethereum blockchain. The contract allows authorized addresses to create proposals, vote on them, and execute proposals when certain conditions are met.

**Contract Structure**

**Proposal Struct**

* **address target**: The address of the contract or account that will be interacted with when the proposal is executed.
* **bytes data**: Encoded data that will be sent to the target contract.
* **uint yesCount**: The number of 'yes' votes the proposal has received.
* **uint noCount**: The number of 'no' votes the proposal has received.
* **bool executed**: A flag indicating whether the proposal has been executed.

**State Variables**

* **Proposal[] public proposals**: An array of all proposals created.
* **uint constant MIN\_YES = 10**: The minimum number of 'yes' votes required for a proposal to be executed.
* **mapping(address => bool) public isAllowed**: A mapping of addresses allowed to interact with the contract.
* **mapping(uint => mapping(address => bool)) public hasVoted**: Tracks whether an address has voted on a specific proposal.
* **mapping(uint => mapping(address => bool)) public voteChoice**: Records the voting choice of each address for each proposal.

**Events**

* **event ProposalCreated(uint proposalID)**: Emitted when a new proposal is created.
* **event VoteCast(uint proposalID, address voter)**: Emitted when a vote is cast on a proposal.
* **event ProposalExecuted(uint proposalID)**: Emitted when a proposal is executed.

**Constructor**

The constructor initializes the contract with a list of allowed addresses, including the address deploying the contract.

solidity

Copy code

constructor(address[] memory allowedAddresses) {

isAllowed[msg.sender] = true;

for (uint i = 0; i < allowedAddresses.length; i++) {

isAllowed[allowedAddresses[i]] = true;

}

}

**Modifiers**

* **onlyAllowed**: Ensures that only addresses marked as allowed can execute functions with this modifier.

**Functions**

* **newProposal(address \_target, bytes calldata \_data)**: Allows an authorized address to create a new proposal. Emits the ProposalCreated event.
* **castVote(uint \_propID, bool \_supports)**: Allows an authorized address to vote on a proposal. If a proposal receives 10 or more 'yes' votes, it is executed. Emits the VoteCast event.
* **executeProposal(uint \_propID)**: Executes a proposal by calling the target address with the specified data. Emits the ProposalExecuted event. This function is called internally when a proposal meets the 'yes' vote threshold.

**Usage**

1. **Deploy** the contract and provide a list of allowed addresses.
2. **Create Proposals**: Use newProposal to add proposals to the system.
3. **Vote**: Authorized addresses can cast votes using castVote.
4. **Execution**: Proposals with 10 or more 'yes' votes will be automatically executed.

**Security**

* Only addresses listed in the isAllowed mapping can create proposals or vote, ensuring controlled access.
* Proposals cannot be executed more than once.