gmx No.5 governor

https://github.com/xiaoyu1998

Resolution:2560×1440(2K)
version: e19b0d00d5bef205a5f86c1606d1aba300990536

Entries

propose
@openzeppelin/contracts/governance/Governor.sol

```
uint256 proposerVotes = getVotes(proposer, clock() - 1);
uint256 votesThreshold = proposalThreshold();
return _propose(targets, values, calldatas, description, proposer);
```

castVote
@openzeppelin/contracts/governance/Governor.sol

```
return _castVote(proposalId, voter, support, "");
```

_castVote
@openzeppelin/contracts/governance/Governor.sol

```
return _castVote(proposalId, account, support, reason, _defaultParams());
```

_castVote
@openzeppelin/contracts/governance/Governor.sol

```
uint256 weight = _getVotes(account, proposalSnapshot(proposalId), params);
_countVote(proposalId, account, support, weight, params);
```

execute
@openzeppelin/contracts/governance/Governor.sol

```
_execute(proposalId, targets, values, calldatas, descriptionHash);
```

queue
@openzeppelin/contracts/governance/extensions/GovernorTimelockControl.sol

```
_timelock.scheduleBatch(targets, values, calldatas, 0, descriptionHash, delay);
```

GovToken

getVotes
@openzeppelin/contracts/governance/Governor.sol

```
return _getVotes(account, timepoint, _defaultParams());
```

_getVotes
@openzeppelin/contracts/governance/extensions/GovernorVotes.sol

```
return token.getPastVotes(account, timepoint);
```

getPastVotes
@openzeppelin/contracts/token/extensions/ERC20Votes.sol

```
return _checkpointsLookup(_totalSupplyCheckpoints, timepoint);
```

_countVote
@openzeppelin/contracts/token/compatibility/GovernorCompatibilityBravo.sol

```
ProposalDetails storage details = _proposalDetails[proposalId];
```

GovernorTimelockControl

_execute
@openzeppelin/contracts/governance/extensions/GovernorTimelockControl.sol

```
_timelock.executeBatch{value: msg.value}(targets, values, calldatas, 0,
descriptionHash);
```

TimelockController

_execute
@openzeppelin/contracts/governance/TimelockControl.sol

```
_beforeCall(id, predecessor);
_execute(target, value, payload);
_afterCall(id);
```

_beforeCall
@openzeppelin/contracts/governance/TimelockControl.sol

```
if (!isOperationReady(id)) {
```

isOperationReady
@openzeppelin/contracts/governance/TimelockControl.sol

```
return getOperationState(id) == OperationState.Ready;
```

getOperationState
@openzeppelin/contracts/governance/TimelockControl.sol

```
uint256 timestamp = getTimestamp(id);
```

scheduleBatch
@openzeppelin/contracts/governance/TimelockControl.sol

```
_schedule(id, delay);
```

scheduleBatch
@openzeppelin/contracts/governance/TimelockControl.sol

```
_timestamps[id] = block.timestamp + delay;
```