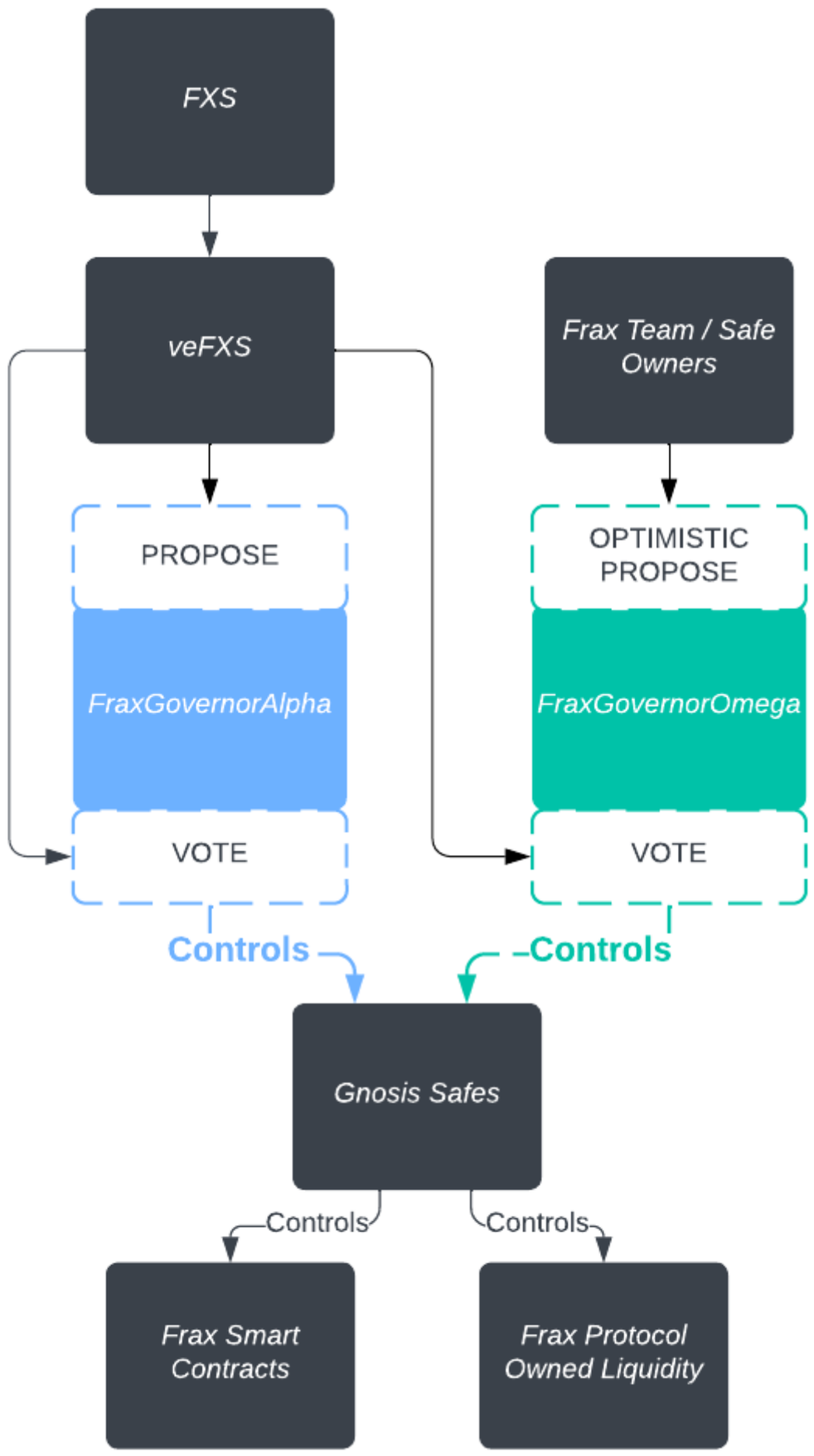
Frax Governance Overview

Fully onchain, trustless, and decentralized governance based on Compound/OpenZeppelin Governor that controls Gnosis Safes

Motivation

Prior to Frax Governance, Frax Protocol operations were not fully decentralized. Most actions were taken by key Frax stakeholders through Gnosis Safes. The implied trust assumption was that the Frax stakeholders won't act maliciously and external actors won't force the stakeholders to execute malicious actions. With the introduction of Frax Governance, this trust assumption is removed and Frax Protocol is controlled by veFXS holders through onchain governance.

Major Components



Frax Governance

End State and Key Takeaways

The final state of Frax Governance will have full control of all major safes, giving veFXS holders the final say over everything in the protocol. veFXS holders can vote for or against all proposals and can replace Frax team members as owners on the underlying Safes and execute any action, if it reaches quorum and passes.

Frax team members have the ability to create optimistic proposals through FraxGovernorOmega which succeed by default. Typically proposals fail by default, but with Frax Governance, veFXS holders can vote for or against optimistic proposals as well. If an optimistic proposal reaches quorum and there are more against than for votes, the proposal will fail.

← Fraxferry - Previous Details

Next - Frax Governance How It Works →

How It Works

Prior Work

Frax Governance is based largely on Compound/OpenZeppelin Governor contracts. The lifecycle of a Governor proposal is covered here: https://docs.openzeppelin.com/contracts/4.x/governance#proposal_lifecycle.

Frax Governance

Frax Governance is a dual Governor system consisting of FraxGovernorAlpha and FraxGovernorOmega. Alpha and Omega have different uses and configurations.

FraxGovernorOmega

FraxGovernorOmega has limited control over the underlying Safes. Only the Frax Team / Safe owners can create Omega proposals. Each proposal is mapped 1 to 1 to a Safe transaction. Omega is configured as an additional Safe owner and must approve of a Safe transaction before Safe owners can execute it. This is enforced using a custom Gnosis Safe Guard (https://docs.safe.global/safe-core-protocol/hooks).

Omega has a very short voting delay, a short voting period, and a low quorum value. Safe owners can only add proposals to Omega after getting the Safe's threshold number of signatures for that Safe transaction. For example, if a Safe is a 3/6 (5 owners + Omega) 3 owners must sign before the proposal can be put into Omega.

After adding a proposal to Omega, it follows the same lifecycle as a Governor proposal. If it passes, Omega calls approveHash on the Safe and the proposal can be executed by any owner through the Gnosis Safe UI as usual. If the proposal fails, rejectTransaction can be called on Omega to approve a 0 ether transfer. Other owners can sign and execute the same 0 ether transfer, which just increments the nonce on the Safe, allowing creation of new Gnosis transactions.

Omega has a feature called the short circuit threshold. **If 51% of the total veFXS supply votes for an Omega proposal, it will immediately succeed and be executable**. This helps in extraordinary circumstances, when the Frax team needs to take quick action to protect the protocol.

Omega cannot change its own governance parameters or change any configuration on the underlying Safes. FraxGovernorAlpha must be used to change these values.

FraxGovernorAlpha

FraxGovernorAlpha has full control over underlying Safes. Any veFXS holder with a veFXS balance meeting or exceeding the proposal threshold can create a proposal. FraxGovernorAlpha is nearly identical to OpenZeppelin Governor. It has full control over each Gnosis Safe because its TimelockController is configured as a module (https://docs.safe.global/safe-core-protocol/plugins) on the Safe and can execute any action. Modules are not constrained by Gnosis Safe Guards.

Alpha has a long voting delay, long voting period, and much higher quorum than Omega. Once an Alpha proposal passes, it must be queued and after the configured timelock period, it can be executed.

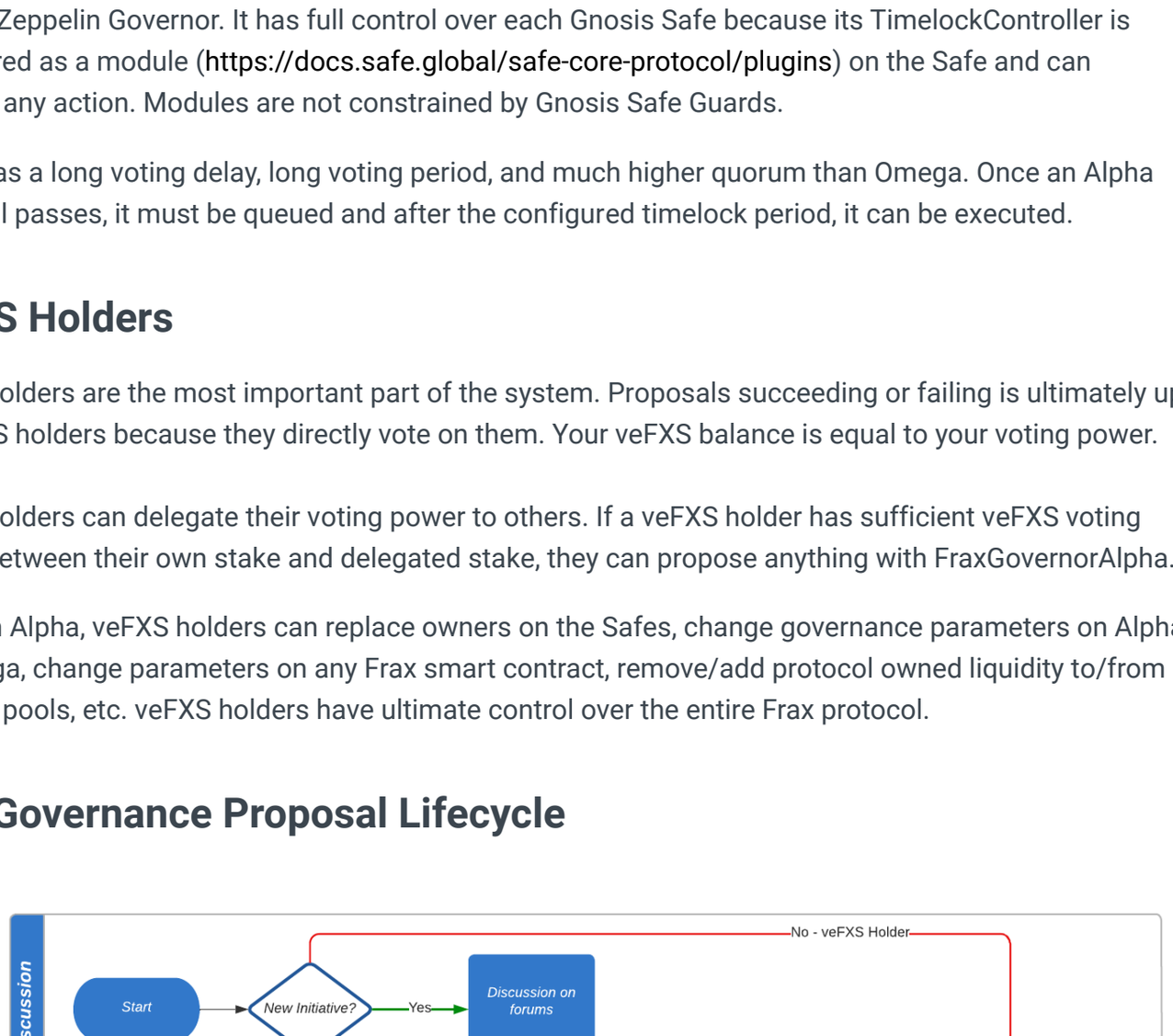
veFXS Holders

veFXS holders are the most important part of the system. Proposals succeeding or failing is ultimately up to veFXS holders because they directly vote on them. Your veFXS balance is equal to your voting power.

veFXS holders can delegate their voting power to others. If a veFXS holder has sufficient veFXS voting power between their own stake and delegated stake, they can propose anything with FraxGovernorAlpha.

Through Alpha, veFXS holders can replace owners on the Safes, change governance parameters on Alpha or Omega, change parameters on any Frax smart contract, remove/add protocol owned liquidity to/from liquidity pools, etc. veFXS holders have ultimate control over the entire Frax protocol.

Frax Governance Proposal Lifecycle



Frax Governance - Previous
Frax Governance Overview

Next - Frax Governance
Advanced Concepts

Advanced Concepts

Delegations

Due to how the veFXS contract works, if you lock more FXS or increase the duration of your lock, you have to delegate your voting weight again using the `VeFxsVotingDelegation` contract.

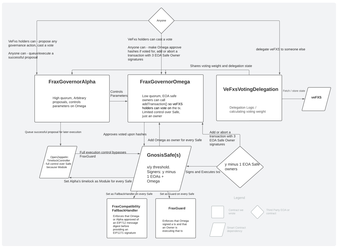
To delegate back to yourself, call the `delegate()` function with your address as the delegatee.

Delegations can also be done by signature. See `delegateBySig()`.

veFXS Voting Weight Decay

When your veFXS lock expires, your balance is equal to the amount of FXS you locked. In Frax Governance, as soon as your lock expires your voting weight goes to 0. You need to have an active lock to have voting weight in Frax Governance.

Smart Contract Design



Create an Alpha proposal

1. Go to https://app.frax.finance/gov/frax
2. Click Propose
3. Make sure your voting weight is above the proposal threshold
    1. Check proposal threshold here: https://etherscan.io/address/0xe8ab863e629a05c73d6a23b99d37027e3763156e#readContract#F25
    2. Check your weight here: https://etherscan.io/address/0x6b83c4f3a6729fb7d5e19b720092162df439f567#readContract#F18
4. Fill out your transaction data
5. Submit

Fractional Voting

Frax Governance makes use of ScopeLift's fractional voting: https://github.com/ScopeLift/flexible-voting/blob/master/src/GovernorCountingFractional.sol

This allows for users to split their voting power between for/against/abstain. It also allows smart contracts/custodians to pass through the voting power to liquidity providers based on their relative LP stake.

Governance Configuration

For a Safe to be properly configured for frxGov, it must have the FraxGuard installed, have FraxGovernorOmega as a signer, have FraxGovernorAlpha's TimelockController installed as a module, and have FraxCompatibilityFallbackHandler set as the fallbackHandler. Safes can be sunset similarly through Alpha proposals by removing the configuration and removing them from the allowlist.

FraxGovernorOmega and FraxGovernorAlpha uses OpenZeppelin Governor and the following extensions:

- GovernorSettings
- GovernorVotesQuorumFraction
- GovernorCountingFractional

In addition, Alpha uses these Governor extensions:

- GovernorTimelockControl
- GovernorPreventLateQuorum

See their configuration here:

https://docs.openzeppelin.com/contracts/4.x/api/governance#GovernorVotesQuorumFraction

https://docs.openzeppelin.com/contracts/4.x/api/governance#GovernorTimelockControl

https://docs.openzeppelin.com/contracts/4.x/api/governance#GovernorSettings

https://docs.openzeppelin.com/contracts/4.x/api/governance#GovernorPreventLateQuorum

Additional Frax Governance Specific Configuration:

All of the following functions must be called from an Alpha proposal.

Alpha and Omega:

- `setVotingDelayBlocks()`
- `setVeFxsVotingDelegation()`
- `updateShortCircuitNumerator()`

Omega only:

- `addToSafeAllowlist()`
- `removeFromSafeAllowlist()`
- `addToDelegateCallAllowlist()`
- `removeFromDelegateCallAllowlist()`
- `setSafeVotingPeriod()`

See the frxGov codebase for explanation of each of these functions.

See the Gnosis Safe codebase for Safe configuration that Alpha controls.

Full Proposal Flow

Alpha

Identical to OpenZeppelin Governor with GovernorTimelockControl.

- Create proposal - `propose()`
- Vote - `castVote()`
- If fails do nothing
- If succeeds
    - `queue()`
    - Wait Timelock's `minDelay`
    - `execute()`

Omega

1. An owner uses the Gnosis Safe to initiate a DeFi transaction. This produces a transaction hash that identifies the action to be approved or rejected by the other Safe owners.
2. ½ EOA sign the transaction through the UI
3. After 3 signatures are collected, anyone can call `fraxGovernorOmega.addTransaction(address teamSafe, TxHashArgs calldata args, bytes calldata signatures)` to begin onchain governance. The team is incentivized to do so because they cannot execute any Gnosis transaction without FraxGovernorOmega's approval.
4. veFXS voters have a 2-day window of time to vote on the proposal.
5. If no quorum is met during the voting window, or there are more for than against votes on the proposal, anyone can call `execute()`. This calls `safe.approveHash()` from Omega under the hood. It provides the needed approval from FraxGovernorOmega, which will allow the gnosis transaction to be executed through the Gnosis UI.
    1. If quorum is met and there are more against than for votes. The proposal gets vetoed by anyone calling `fraxGovernorOmega.rejectTransaction (address teamSafe, uint256 nonce)`. This will cause FraxGovernorOmega to sign a zero ETH transfer Safe transaction with the same nonce. Safe owners can then sign the same transaction in the UI using the "replace transaction" functionality. Safe owners can then execute the zero ETH transfer, incrementing the nonce. The original transaction can never be executed, because there is no approval from FraxGovernorOmega and the nonce has moved on, invalidating the original transaction.

Smart Contract Signature Support (EIP1271 / EIP712)

Background: https://help.safe.global/en/articles/40783-what-are-signed-messages

Gnosis Safes can provide smart contract signatures. The typical mechanism is contained in Safe's CompatibilityFallbackHandler, which checks if the threshold number of signatures is met from the owners, and then the Safe provides its approval. This mechanism was not sufficient for frxGov because we want Omega + owners to approve and OR Alpha alone to provide approval.

To solve this we wrote FraxCompatibilityFallbackHandler, which only checks if `safe.signedMessages(messageHash)` is set. Alpha and Omega can both call the setter of this state using `SignMessageLib.signMessage()`.

Attacks and Mitigations

FraxGovernorAlpha could be used to create a proposal that essentially shuts down the entire Frax protocol and returns all protocol owned liquidity to FXS holders. We have the Timelock delay on Alpha for situations like this. It will take a day from when this proposal passes to when it can be executed, giving Frax users time to exit the protocol via various liquidity pools.

We also added in the OpenZeppelin Governor extension GovernorPreventLateQuorum. If a proposal reaches quorum late in the voting period, the voting period will get extended by 2 days. This stops malicious user(s) from voting for a malicious proposal at the last second. It gives time for the community to react in such a scenario.

We also created the frxEth redemption queue to allow users to exit frxEth if a malicious actor gets 51% of veFXS voting power and passes a proposal to infinite mint frxEth in an attempt to steal the treasury.

Technical Discussions

To conform to the ERC5805 standard, an implementing contract must implement several functions including `getPastTotalSupply(uint256 timepoint) external view returns (uint256)`. We use timestamps throughout our Governor contracts, however this function is the one exception and it must accept a `block.number`. Unfortunately, `veFXS.totalSupply(timestamp)` doesn't work for historical values, so we must use `veFXS.totalSupplyAt(block.number)`.

This also impacts quorum values. Typically quorum is calculated from voting start timestamp (snapshot). We allow similar functionality by making the `$votingDelayBlocks` value configurable by governance, so it mirrors the timestamp functionality.

Codebase and Contract Addresses

Codebase

https://github.com/FraxFinance/frax-governance

Contract Addresses

VeFxsVotingDelegation: https://etherscan.io/address/0x6b83c4f3a6729fb7d5e19b720092162df439f567

FraxGovernorAlpha: https://etherscan.io/address/0xe8Ab863E629a05c73D6a23b99d37027E3763156e

FraxGovernorAlphaTimelock: https://etherscan.io/address/0x821794E69d2831975a11f80E8092c682D5Ec8F83

FraxGovernorOmega: https://etherscan.io/address/0x953791D7CSac8Ce5fb23BBBF88963DA37a95FE7a

FraxGuard: https://etherscan.io/address/0xed53eb15b2011395A7353e076024CBC9F19481D0

FraxCompatibilityFallbackHandler: https://etherscan.io/address/0x3FeFB779d737aCEa272686eA6E174ebF4273F973