Jailed validators

Validators can bejailed for several reasons, such as double signing a block, signing an invalid block, or excessive downtime. Each of these reasons will have different ramifications for the validator.

When a validator is jailed, it is removed from the validator sets and cannot participate in consensus. The validator node can still run and process the new blocks. A jailed validator can also still receive bonds.

When the protocol determines that a validator will be jailed, the jailing and validator set removal will occur at the beginning of the next new epoch.

You can check if a validator is jailed by querying its state with the following command:

namadac

validator-state

--validator

< validator_addres s

Unjailing a validator

Once jailed, validators remain jailed indefinitely. They can only be unjailed by anunjail-validator transaction using the validator's signing keys. This can be done with the following command:

namadac

unjail-validator

--validator

< validator_addres s

> Because the validator alias sometimes clashes with the alias for the implicit account and or established account in the wallet, it is recommended to use the validator address instead of the alias. In order to find the validator address, you can use the following command:

namadaw

list

--addr

|

grep

< validator-alia s

> then make sure it is the correspondingtnam address. If the transaction is successful, then the validator will be reinstated into one of the validator sets at the pipeline length relative to the current epoch (typically 2 epochs in the future).

There may be certain restrictions on unjailing your validator depending on the original reason for jailing. These will be described below.

Jailing for protocol misbehavior

Protocol misbehaviors include the aforementioned double signing of a block and signing of an invalid block.

In these cases, the validator will also be slashed - it and its delegates will lose stake and voting power.

When a slash is detected and enqueued by the protocol, it is scheduled to be processedunbonding_len + cubic_slashing_window_len + 1 epochs after the infraction epoch. This is to allow for sufficient time to detect all possible correlated infractions to be considered for cubic slashing.

The validator will also be considered to befrozen until it no longer has any unprocessed, enqueued slashes. Whilefrozen , unbonding from the validator is prohibited, and unjailing is forbidden.

A validator's slash history, with previously processed slashes and enqueued slashes for future processing, can be queried with the following command:

namadac

slashes

--validator

< validator_addres s

Additionally, all slashes in the network can be queried without the--validator flag.

Jailing for downtime (liveness)

While a validator is in the consensus set, its uptime and voting history are tracked to ensure sufficient liveness. Two proof-of-stake parameters are used to determine if a validator will be jailed for liveness:

- liveness_window_check
- liveness_threshold

Namada liveness requires that a consensus validator vote on at least aliveness_threshold fraction of the lastliveness_window_check blocks. If at any point this fraction dips below theliveness_threshold , then the validator will be jailed.

The values of these parameters can be queried with the command:

namadac

query-protocol-parameters Since jailing for downtime occurs without the presence of slashes (they are notfrozen), the jailed validator can unjail themselves as soon as they become jailed.

Proof of stake Starting a network