Network Nervous System

The NNS is the decentralized autonomous organization (DAO) that governs the Internet Computer (IC). It is a fully on-chain, decentralized system and is, for instance, responsible for making protocol level upgrades to continuously improve the Internet Computer. It does this via an implementation of liquid democracy in which ICP neuron holders vote on proposals that shape the development of the Internet Computer. Once such a proposal is accepted, it is autonomously executed.

While other blockchains take weeks or months to upgrade (sometimes called hard fork) and typically require substantial manual work and coordination to do so, the IC upgrades itself on a weekly basis (https://dashboard.internetcomputer.org/releases). Its ability to upgrade and iterate is a comparative "superpower."

The NNS works by allowing users to stake ICP governance tokens to create voting neurons. Anyone can create a neuron, which together exert the will of the community, mediated through algorithms. Neurons are like accounts where notice of withdrawals must be given, where the length of the notice period is a configuration known as the “dissolve delay.” The voting power of neurons, and their relative claim to voting rewards, is proportional to the quantity of staked ICP, the length of the configured dissolve delay, and their “age.” Neurons can be made to vote manually, or automatically, by following other neurons in a form of liquid democracy.

Neuron holders are placed in a cryptoeconomic game, where they are incentivized to vote to adopt and reject proposals, or to configure neuron follows that cause them to vote automatically in a desirable way, according to what is most likely to drive the value of the Internet Computer network over the long term.

This is the first time in history when a decentralized infrastructure will self-direct with the aim of competing with proprietary centralized infrastructures run by commercial organizations with leaders and boards.

Overview

The purpose of the NNS is to allow the Internet Computer network to be governed in an open, decentralized, and secure manner. It has complete control over all aspects of the network. For example, it can upgrade the protocol and software used by the node machines that host the network; it can create new subnet blockchains to increase network capacity; it can split subnets to divide their load; it can configure economic parameters that control how much must be paid by users for compute capacity; and, in extreme situations, it can freeze malicious canister smart contract software in order to protect the network; and many other things. The NNS works by accepting proposals, and deciding to adopt or reject them based on voting activity by “neurons” that network participants have created.

Neurons are also used by participants to submit new proposals. After submission, proposals are either adopted or rejected, which can happen almost immediately, or after some delay, depending upon how the totality of neurons vote. Each proposal is an instance of a specific “proposal type,” which determines what information it contains. For each type of proposal, the NNS maintains a corresponding system function, which it invokes whenever a proposal of that type is adopted. When a proposal is adopted by the NNS, it invokes the corresponding system function by drawing information from the proposal’s content to fill the parameters. Each type of proposal belongs to a specific “proposal topic,” such as “#NodeAdmin” or “#NetworkEconomics,” which determines details about how it will be processed.To prevent users (neurons) from spamming the NNS with proposals, a fee is levied on the neuron that submitted a proposal if it is rejected.

The NNS decides whether to adopt or reject proposals by watching how neurons emit votes. Anyone can create a neuron by locking balances of the Internet Computer’s native utility token (ICP) that is hosted on a ledger inside the NNS. When a user creates a neuron, the locked balance of ICP can only be unlocked by disbursing (“destroying”) the neuron. Users are incentivized to create neurons because they earn rewards when they vote on proposals. Rewards take the form of newly minted ICP that are created by the NNS. The quantity of ICP rewards disbursed to a neuron derive from such factors as the size of the locked balance, the minimum lockup period remaining (the “dissolve delay”), the neuron’s “age,” the proportion of possible votes it has participated in, and the sum of voting activity across all neurons, since the overall total rewards disbursed is capped and must be divided between voters.

At any moment, each neuron has a currently configured “dissolve delay.” This determines how long it will take to dissolve if it is placed into “dissolve mode.” Once a neuron has been placed into “dissolve mode,” its dissolve delay falls over the passage of time, rather like a kitchen timer, until it reaches zero, whereupon its owner can perform a final disburse action to unlock the balance of ICP. The dissolve delay creates an economic incentive for neuron owners to vote with a view toward maximizing the value of their locked ICP balances at a future date. ICP is a proxy for the success of the network over the long term, sans short-term volatility, this creates an economic incentive to vote in the best interests of the network. Neuron owners can freely configure higher dissolve delays, up to a maximum delay of eight years, but cannot decrease the dissolve delay other than by the natural passage of time. The NNS pays higher voting rewards the higher the dissolve delay, encouraging users to enter a game in which an economic incentive is created to vote according to a very long-term vision.

Neuron owners may find it hard to manually vote directly on every proposal submitted to the NNS. Firstly, large volumes of proposals may be submitted to the NNS, often at awkward times, and owners may not be available or have the necessary time to evaluate each one. Secondly, neuron owners may lack the necessary expertise to evaluate proposals themselves. The NNS uses a form of liquid democracy to address these challenges. For any proposal topic, a neuron can be configured to vote automatically by following the votes of a group of neurons, voting to adopt proposals whenever more than half of the followees vote to adopt, and voting to reject whenever at least half of the followees vote to reject. A catch-all follow rule may also be defined to make a neuron vote automatically on proposals with topics for which no follow rule has been defined. It is assumed that neuron owners will manage how their neurons follow other neurons in the best interests of the network, which is also in their own economic interests, owing to their locked ICP balances.

It is expected that a large proportion of the overall supply of ICP will be locked in order to earn rewards. This secures the Internet Computer’s network governance, by making it both difficult and exorbitantly expensive for an attacker to acquire a sufficiently large stake to gain significant influence. Since neuron owners may wish to maximize their rewards by voting on all proposals, most neurons will either be actively managed or configured to follow other neurons so that they can vote automatically. In practice, once trusted neurons have voted on proposals, a majority of the other neurons will also vote as the result of cascading follow relationships. This means that the NNS can usually quickly determine whether a majority of the overall voting power represented by all neurons wishes to adopt or reject a proposal, and decide on the proposal accordingly. However, the NNS cannot depend on obtaining such a majority since, in principle, neuron owners may not define follow rules, or they may simply choose not to vote.

Proposals

Format

Each proposal submitted to the NNS has the following fields:

- **Summary:** Text providing a short description of the proposal, composed using a maximum of 280 bytes.
- **URL:** The web address of additional content required to evaluate the proposal, specified using HTTPS. For example, the address might describe content supporting the assignment of a DCID (data center id) to a new data center.
- **Proposer:** The ID of the neuron that submitted the proposal. When a proposal is submitted, a “charge” is placed on its balance in case it is rejected. So the balance needs to be big enough to pay the charge on (all) rejection(s). A neuron is required to have a dissolve delay ≥ 6 months to vote, and this applies to submitting proposals too.
- **Proposal Type:** The type of the proposal. This infers what topic it belongs to (e.g., #NodeAdmin), the system function that will process the proposal if it is adopted, and the type and structure of the parameters that will be passed to that function.
- **Parameters:** The parameters that will be passed to the system function that will be invoked if the proposal is adopted, as determined by its type. When a proposal is submitted, the NNS checks the parameters.

The NNS assigns a unique identity to each proposal that it receives.

The NNS assigns a unique identity to each proposal that it receives.

Topics

The topic of a proposal, which is inferred from its type, determines how it will be processed. For example, the NNS may require voters to have a greater degree of agreement, or to try to process proposals faster, for some topics. Also, neurons follow other neurons on a per-topic basis. Initial topics include:

- **#NeuronManagement:** A special topic by means of which a neuron can be managed by the followees for this topic (in this case, there is no fallback to default). Votes on this topic are not included in the voting history of the neuron. For proposals on this topic, only the neuron’s followees on the topic that the proposals pertain to are allowed to vote. Because the set of eligible voters of proposals on this topic is restricted, proposals on this topic have a shorter than normal voting period.
- **ExchangeRate:** All proposals provide information in “real time” about the market value of ICP, as measured by an International Monetary Fund (IMF) Special Drawing Right (SDR) , which allows the NNS to convert ICP to cycles (which power computation) at a rate that keeps their real-world cost constant. Because proposals on this topic are very frequent, they have a shorter voting period, and votes on this topic are not included in the voting history of the neuron.
- **#NetworkEconomics:** Proposals that administer network economics — for example, determining what rewards should be paid to node operators.
- **#Governance:** All proposals that administer governance — for example, motions and the configuration of certain parameters.
- **#NodeAdmin:** All proposals that administer node machines somehow, including but not limited to upgrading or configuring the OS, upgrading or configuring the virtual machine framework, and upgrading or configuring the node replica software.
- **#ParticipantManagement:** All proposals that administer network participants — for example, granting and revoking DCIDs (data center identities) or NPIDs (node provider identities).
- **#SubnetManagement:** All proposals that administer network subnets — for example, creating new subnets, adding and removing subnet nodes, and splitting subnets.
- **#NetworkCanisterManagement:** Installing and upgrading “system” canisters that belong to the network — for example, upgrading the NNS.
- **#KYC:** Proposals that update KYC information for regulatory purposes — for example, during the initial Genesis distribution of ICP in the form of neurons.
- **#NodeProviderRewards:** Topic for proposals to reward node providers.

Types

Initial proposal types include:

- **ManageNeuron (#NeuronManagement, Restricted Voting)** This type of proposal calls a major function on a specified target neuron. Only the followees of the target neuron may vote on these proposals, which effectively provides the followees with control over the target neuron. This can provide a convenient and highly secure means for a team of individuals to manage an important neuron. For example, a neuron might hold a large balance, or belong to an organization of high repute, and be publicized so that many other neurons can follow its vote. In both cases, managing the private key of the principal securely could be problematic. (Either a single copy is held, which is very insecure and provides for a single party to take control, or a group of individuals must divide responsibility — for example, using threshold cryptography, which is complex and time consuming). To address this using this proposal type, the important neuron can be configured to follow the neurons controlled by individual members of a team. Now they can submit proposals to make the important neuron perform actions, which are adopted if and only if a majority of them vote to adopt. (Submitting such a proposal costs a small fee, to prevent denial-of-service attacks.) Nearly any command on the target neuron can be executed, including commands that change the follow rules, allowing the set of team members to be dynamic. Only the final step of dissolving the neuron once its dissolve delay reaches zero cannot be performed using this type of proposal, since this would allow control/“ownership” over the locked balances to be transferred. (The only exception to this rule applies to not-for-profit organizations, which may be allowed to dissolve their neurons without using the initial private key.) To prevent a neuron falling under the malign control of the principal’s private key by accident, the private key can be destroyed so that the neuron can only be controlled by its followees, although this makes it impossible to subsequently unlock the balance.

- **ManageNetworkEconomics (#NetworkEconomics)** This is a single proposal type which can update one or several economic parameters:
  - Reject cost: The amount of ICP the proposer of a rejected proposal will be charged — to prevent the spamming of frivolous proposals.
  - Minimum Neuron Stake: Set the minimum number of ICP required for creation of a neuron. The same limit must also be respected when increasing dissolve delay or changing the neuron state from dissolving to aging.
  - Neuron Management fee: The cost in ICP per neuron management proposal. Here the NNS is doing work on behalf of a specific neuron, and a small fee will be applied to prevent overuse of this feature (i.e., spam).
  - Minimum ICP/SDR rate: To prevent mistakes, there is a lower bound for the ICP/SDR rate, managed by network economic proposals.
  - Dissolve delay of spawned neurons: The dissolve delay of a neuron spawned from the maturity of an existing neuron.
  - Maximum node provider rewards: The maximum rewards to be distributed to node providers in a single distribution event (proposal).
  - Transaction fee: The transaction fee that must be paid for each ledger transaction.
  - Maximum number of proposals to keep per topic: The maximum number of proposals to keep, per topic. When the total number of proposals for a given topic is greater than this number, the oldest proposals that have reached a “final” state may be deleted to save space.
- **Motion (#Governance)** A motion is a text that can be adopted or rejected. No code is executed when a motion is adopted. An adopted motion should guide the future strategy of the Internet Computer ecosystem.
- **ApproveGenesisKYC (#KYC)** When new neurons are created at Genesis, they have GenesisKYC=false. This restricts what actions they can perform. Specifically, they cannot spawn new neurons, and once their dissolve delays are zero, they cannot be disbursed and their balances unlocked to new accounts. This proposal sets GenesisKYC=true for batches of principals.

(Special note: The Genesis event disburses all ICP in the form of neurons, whose principals must be KYCed. Consequently, all neurons created after Genesis have GenesisKYC=true set automatically since they must have been derived from balances that have already been KYCed.)

- **AddOrRemoveNodeProvider (#Participant Management)** Assign (or revoke) an identity to a node provider, associating key information regarding the legal person associated that should provide a way to uniquely identify it.
- **RewardNodeProvider (#NodeProviderRewards)** Propose to reward a Gen-1 node provider an amount of ICP as compensation for providing Gen-1 nodes to the IC.
- **SetDefaultFollowees (#Governance)** Specify the list of followees that a freshly created neuron should have.

The following is a list of proposal types that call other NNS canisters:

- **CreateSubnet (#SubnetManagement)** Combine a specified set of nodes, typically drawn from data centers and operators in such a way as to guarantee their independence, into a new decentralized subnet. The execution of this external update first initiates a new instance of the distributed key generation protocol. The transcript of that protocol is written to a new subnet record in the registry, together with initial configuration information for the subnet, from where the nodes comprising the subnet pick it up.
- **AddNodeToSubnet (#SubnetManagement)** Add a new node to a subnet. The node cannot be currently assigned to a subnet. The execution of this proposal changes an existing subnet record to add a node. From the perspective of the NNS, this update is a simple update of the subnet record in the registry.

- **InstallNetworkCanister (#NetworkCanisterManagement)** A proposal to add a new canister to be installed and executed in the NNS subnetwork. The root canister, which controls all canisters on the NNS except for itself, handles this proposal type. The call also expects the Wasm module that shall be installed.
- **UpgradeNetworkCanister (#NetworkCanisterManagement)** A proposal to upgrade an existing canister in the NNS subnetwork. This proposal type is executed by the root canister. Beyond upgrading the Wasm module of the target canister, the proposal can also set the authorization information and the allocations.
- **BlessReplicaVersion (#NodeAdmin)** A proposal to bless a new version to which the replicas can be upgraded. The proposal registers a replica version (identified by the hash of the installation image) in the registry. Besides creating a record for that version, the proposal also appends that version to the list of “blessed versions” that can be installed on a subnet. By itself, this proposal does not effect any upgrade. (In the future, there will only be one blessed version of the replica software at any given time.)
- **RecoverSubnet (#SubnetManagement)** Update a subnet’s recovery CUP (used to recover subnets that have stalled). Nodes that find a recovery CUP for their subnet will load that CUP from the registry and restart the replica from that CUP.
- **UpdateSubnetConfig (#SubnetManagement)** Update a subnet’s configuration. This proposal updates the subnet record in the registry, with the changes being picked up by the nodes on the subnet when they reference the respective registry version. Subnet configuration comprises protocol parameters that must be consistent across the subnet (e.g., message sizes).
- **AssignNPID (#ParticipantManagement)** Assign an identity to a node operator associating key information regarding its ownership, the jurisdiction in which it is located, and other information. The node operator is stored as a record in the registry. It contains the remaining node allowance for that node operator, that is the number of nodes the node operator can still add to the IC. When an additional node is added by the node operator, the remaining allowance is decreased.
- **RootUpgrade (#NetworkCanisterManagement)** A proposal to upgrade the root canister in the NNS subnetwork. The proposal is processed by the Lifeline canister, which controls the root canister. The proposal updates the Wasm module as well as the authorization settings.
- **SetICPSDR (#ExchangeRate)** Instruct the NNS about the market value of 1 ICP as measured by an IMF SDR. This setting affects cycles pricing (as the value of cycles shall be constant with respect to IMF SDRs).
- **UpgradeSubnetToReplicaVersion (#SubnetManagement)** Update the replica version running on a given subnet. The proposal changes the replica version that is used on the specified subnet. The version must be contained in the list of blessed replica versions. The upgrade is performed when the subnet creates the next regular CUP.
- **ClearProvisionalWhitelist (#NetworkEconomics)** Clears the provisional whitelist, which allows the listed principals to create canisters with cycles. The mechanism is only needed for bootstrap and testing and must be deactivated afterward.
- **RemoveNodeFromSubnet (#SubnetManagement)** Remove a node from a subnet. It then becomes available for reassignment. The execution of this proposal changes an existing subnet record to remove a node. From the perspective of the NNS, this update is a simple update of the subnet record in the registry.
- **SetAuthorizedSubnetworks (#Governance)** Informs the cycles minting canister that a certain principal is authorized to use certain subnetworks (from a list). Can also be used to set the “default” list of subnetworks that principals without special authorization are allowed to use.
- **SetFirewallConfig (#SubnetManagement)** Change the Firewall configuration in the registry (configures which boundary nodes subnet blockchain replicas will communicate with).
- **UpdateNodeOperatorConfig (#NodeAdmin)** Change a node operator’s allowance in the registry.
- **StopOrStartNNSCanister (#NetworkCanisterManagement)** Stop or start an NNS canister.

ICP tokens

ICP are native utility tokens that play three key roles in the network:

1. **Facilitating Network Governance** ICP tokens can be locked to create neurons that participate in network governance by voting, through which they can earn economic rewards.

1. **Production of Cycles for Compute** ICP provides a source store of value that can be converted into “cycles,” which power computation in the role of fuel that is burned when it is used. The NNS converts ICP to cycles at a variable rate, so chosen to ensure users of the network can always create new cycles at approximately constant cost in real terms, such that the cost of acquiring fuel is predictable.

1. **Rewarding Participants** The network mints new ICP to reward and incentivize those playing important roles that enable the network to function, including: a) the provision of “voting rewards” to those participating in governance, and b) the provision of “node provider rewards” to those operating the node machines that are hosting the network.

Ledger

The ICP ledger is hosted within the NNS, and records all balances of ICP in the manner of a spreadsheet. Each row is called an “account,” which has two fields (i.e., there are two “columns”):

1. **Account identifier (bytes)** A unique value that is derived from the identity of the “principal” that “controls” the account. Currently, the principal must either be: (i) the owner of a public key pair, or (ii) a canister smart contract that is part of the NNS. Account identifiers are derived by hashing the concatenation of a domain separator, the principal ID, and the subaccount (or zeros if no subaccount is given).

1. **Balance** (positive integer, representing one hundredth of a millionth of an ICP) The quantity of ICP assigned to the principal of the account.

When the principal is a public key or Canister, they can apply the following operation to an account:

1. **Send** Send a portion of the ICP balance to another account. If all the ICP is sent to another account, then the sending account ceases to exist (i.e., is deleted from the ledger).
2. **Notify** When the destination of the funds sent is the account of an NNS canister (e.g., an account of the governance canister), the sender can ask the ledger to notify the recipient canister of the incoming transfer. The recipient canister can then act on this notification. Two examples where this ability is used are creating a neuron and refreshing the stake of a neuron. These are detailed below.

Operations that require interaction between the ledger and the governance system (Neurons):

1. **Create neuron** When the principal is a public key holder, they may lock a portion of their balance inside a new neuron. Technically, creating the neuron is done in two stages. First transfer the ICP to be staked to an account of the governance canister (which corresponds to a new neuron — the details of the association are omitted here). Then notify the governance canister of the incoming transfer which updates its internal neuron bookkeeping. If the entire balance is locked inside a new neuron, the account ceases to exist (i.e., is deleted from the ledger). To move these ICP to a different account, such as back to the original account, where they can once again be controlled like a normal balance, the associated neuron must be fully dissolved and disbursed (destroyed). The new neuron that has been created is controlled by the private key of the principal that created it.
2. **Refresh stake** The stake of a neuron may be increased by transferring to its address/account in the ledger and notifying the governance canister of the incoming transfer. Refreshing the stake will change the maturity and age of the neuron prorated. For example, if the stake is doubled, the maturity and age will be halved, so spawning will yield the same amount and the age bonus will be the same as before (in absolute terms).

Neurons

A neuron locks a balance of ICP and enables its owner to participate in network governance, through which they can earn rewards.

Attributes

Neurons have the following attributes:

- **Identity (uint64)** The general identity of the neuron object. When a neuron is configured to follow another neuron, this is the value that is used. This is a random 64-bit value selected at neuron creation.
- **Account (bytes, private)** The ledger account where the locked ICP balance resides.
- **Controller (principal ID, private)** The principal that actually controls the neuron. The principal must identify a public key pair, which acts as a “master key,” such that the corresponding secret key should be kept very secure. The principal might control many neurons.
- **Hot Keys (list of principal ID, private)** Keys that can be used to perform actions with limited privileges, such as voting, without exposing the secret key corresponding to the principal (e.g., could be a WebAuthn key).
- **CreatedAt (timestamp)** When the Neuron was created.
- **AgingSince (timestamp)** The timestamp corresponding to the time this neuron has started aging. This is either the creation time or the last time at which the neuron has stopped dissolving. This value is meaningless when the neuron is dissolving, since a dissolving neuron always has an age of zero.

- **DissolveState** At any time, at most one of WhenDissolved and DissolveDelay are specified.
  - WhenDissolved (timestamp)

When the dissolve timer is running, this stores the timestamp in seconds from the Unix epoch, at which point the neuron becomes dissolved. At any time while the neuron is dissolving, the neuron owner may pause dissolving, in which case DissolveDelay will get assigned to: WhenDissolved minus the timestamp when the action is taken.

-
  - DissolveDelay (duration)

When the dissolve timer is stopped, this stores how much time the dissolve timer will be started with. It can be eight years at most. At any time while in this state, the neuron owner may (re)start dissolving, in which case WhenDissolved will get assigned to the timestamp when the action is taken plus DissolveDelay.

- **Maturity (positive number)** The maturity of a neuron which reflects the amount of voting rewards allocated to a neuron. When new neurons are created, their maturity is zero. When neurons vote, over time the NNS increases their maturity to reward them.
- **Follow Relationships (mapping from topic to list of followees, private)** A neuron can be configured to vote automatically by following other neurons on a topic-by-topic basis. For any valid topic, a list of followees can be specified, and the neuron will follow the vote of a majority of the followees on a proposal with a type belonging to that topic. If a null topic is specified, this acts as a catch-all that enables the neuron to follow the vote of followees where a rule has not been specified.
- **Recent Votes (public)** A record of recent votes is maintained. This can provide a guide for those wishing to evaluate whether to follow a neuron, or how their followees are voting.
- **NotForProfit (boolean)** Whether this neuron is “not for profit,” making it dissolvable by voting.

The following attributes can be computed:

- **Age (seconds)** (computed from AgingSince and current time) The period of time that has elapsed since the neuron was created or last stopped dissolving. Conceptually, whenever a neuron starts dissolving, then its age is reset to zero and remains zero while it is dissolving. If a dissolving neuron has dissolving turned off, the current time becomes the effective neuron creation date for the purposes of calculating the age.
- **State (LOCKED or DISSOLVING or DISSOLVED)** (computed from DissolveState and the current time)
  - LOCKED: In this state, the neuron is locked with a specific DissolveDelay. It accrues age by the passage of time and it can vote if DissolveDelay is at least six months. The method start_dissolving can be called to transfer the neuron to the DISSOLVING state. The method increase_dissolve_delay can be used to increase the dissolve delay without affecting the state or the age of the neuron.
  - DISSOLVING: In this state, the neuron’s effective dissolve delay decreases with the passage of time. While dissolving, the neuron’s age is considered zero. Eventually it will reach the DISSOLVED state. The method stop_dissolving can be called to transfer the neuron to the LOCKED state, and the neuron will start aging again. The method increase_dissolve_delay can be used to increase the dissolve delay, but this will not stop the timer or affect the age of the neuron.
  - DISSOLVED: In the dissolved state, the neuron’s stake can be disbursed using the disburse method. It cannot vote as its dissolve delay is considered to be zero. If the method increase_dissolve_delay is called in this state, the neuron will become locked with the specified dissolve delay and start aging again. Neuron holders have an incentive not to keep neurons in the dissolved state for a long time: if the holders want to make their tokens liquid, they disburse the neuron’s stake, and if they want to earn voting rewards, they increase the dissolve delay. If these incentives turn out to be insufficient, the NNS may decide to impose further restrictions on dissolved neurons.
- **ControlByProposals (boolean)** (true if the neuron has a non-empty list of followees on the #NeuronManagement topic) If a neuron specifies followees on the ManageNeuron topic, it can be managed by proposals of type ManageNeuron (#NeuronManagement), which may only be voted upon by the neuron’s own followees. This provides a foundation for the management of highly security sensitive neurons, since it allows them to be maintained without hot keys or the secret key of the principal, which can be kept in cold storage or even destroyed (so long as the associated balance of ICP need never be unlocked). For example, the DFINITY Foundation or the Internet Computer Association might publicize the address of special neurons that will be made to vote according to their wishes, so that others can configure their neurons to follow them and leverage their expertise and efforts in governance. One problem with such practices is that they introduce the risk that secret keys used in the management of the publicized neurons might be compromised, allowing hackers to take control and “trick” large numbers of following neurons into voting according to their wishes. If the publicized neurons have admin proposals enabled, however, then they can be administered by the neurons they follow (their followees), which are typically controlled by

a large number of team members who cannot be simultaneously extorted, without any need for the usage of secret keys whatsoever.

Commands

The principal that controls a neuron may instruct it to perform the following actions:

- **Start Dissolving** The dissolve delay is like a kitchen timer that can only be turned in one direction. It can be arbitrarily increased, but only reduced by turning on dissolve mode and counting down. The neuron can be instructed to start “dissolving.” When the neuron is dissolving, its dissolve delay falls over the passage of time until either it is stopped or it reaches zero. A neuron cannot vote (or earn rewards for voting) when its dissolve delay falls below six months. Once the dissolve delay reaches zero, it stops falling and the controlling principal can instruct the neuron to disburse.
- **Stop Dissolving** A neuron that is dissolving can be instructed to stop, whereupon its dissolve delay stops falling with time.
- **Disburse** When the dissolve delay of the neuron is 0, its controlling principal can instruct it to disburse the neuron’s stake. Its locked ICP balance is transferred to a specified new ledger account, and the neuron and its own ledger account disappear.
- **Increase Dissolve Delay** The dissolve delay of a neuron can be increased up to a maximum of eight years.
- **Spawn** When the maturity of a neuron has risen above a threshold, it can be instructed to spawn a new neuron. This creates a new neuron that locks a new balance of ICP on the ledger. The new neuron can remain controlled by the same principal as its parent, or be assigned to a new principal. When a neuron spawns a new neuron, its maturity falls to zero.
- **Add Hot Key** Add a new hot key that can be used to manage the neuron. This provides an alternative to using the principal’s cold key to manage the neuron, which might be onerous and difficult to keep secure, especially if it is used regularly. A hot key might be a WebAuthn key that is maintained inside a user device, such as a smartphone.
- **Remove Hot Key** Remove a hot key that has been previously assigned to the neuron.

The following actions can be initiated using the principal or a hot key that has been configured:

- **Vote** Have the neuron vote to either adopt or reject a proposal with a specified ID.
- **Follow** Add a rule that enables the neuron to vote automatically on proposals that belong to a specific topic, by specifying a group of followee neurons whose majority vote is followed. The configuration of such follow rules can be used to: a) distribute control over voting power amongst multiple entities, b) have a neuron vote automatically when its owner lacks time to evaluate newly submitted proposals, c) have a neuron vote automatically when its own lacks the expertise to evaluate newly submitted proposals, and d) for other purposes. A follow rule specifies a set of followees. Once a majority of the followees votes to adopt or reject a proposal belonging to the specified topic, the neuron votes the same way. If it becomes impossible for a majority of the followees to adopt (for example, because they are split 50–50 between adopt and reject), then the neuron votes to reject. If a rule is specified where the proposal topic is null, then it becomes a catch-all follow rule, which will be used to vote automatically on proposals belonging to topics for which no specific rule has been specified. If the list of followees is empty, this effectively removes a follow rule.

See Also

- **The Internet Computer project website (hosted on the IC): internetcomputer.org (https://internetcomputer.org/)**