[

1920×1419 98.5 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/72f19b5f0fa602a24de543e02c8fce7cbad896b5.jpeg)

For the initial inputs, assumptions, and goals, please, refer to theRFP.

Summary (TL;DR)

The “Slow Soul Upgrade Mechanism” offers a mechanism for upgrades both for node software and the chain smart contracts updates. It starts with covering general assumptions as well as all the affiliated parties that are interested in the upgrade solution making and upgrade execution. It continues with the mechanism for deciding whether to upgrade or not. The mechanism is split into two cases: “Node software upgrade” and “Smart contracts and bridge upgrades.” The mechanism relies on Soul Bound Voting, an open stage for community discussions and improvements, and efforts to align all the interested parties while preserving options for critical disagreements in the forms of forks or exits. In Part 2, we describe the voting mechanism in detail. In Part 3, we describe the specific upgrade execution mechanism. In Part 4, we cover different fork scenarios, ensuring the community has an opportunity to move forward with a forked chain whenever and in any configuration they want. At the end, we cover questions that suggest further work and research to handle them.

Comparisons

Below, in the “Details” section, I tried to explain the choice of each decision and why it fits. In this section, I’ll briefly compare the chosen option, SBT, with the most obvious.

The first, most obvious option – is using a governance token. But then, firstly, we depend on token allocation and future token flow. Secondly, we can’t manage the token ownership. That is, for example, be sure that there are different interested parties behind different entities. Both issues will make the upgrade mechanism less robust and predictable.

The second most obvious option – is going in a more centralized way, where upgrade decisions are made by the chain team or the chain team + some committee of trusted representatives (that can be selected by the community or assigned in another way). The part of the community that disagrees with the update will still have an opportunity to fork or just not upgrade. However, this approach contradicts the chain’s chosen path of decentralization (as I can assume from sticking to decentralized sequencing and proving discourse). Furthermore, it will increase the probability of chain splits as a chain fork will be the only option for the community to stick to their needs and values without the opportunity to discuss and collaborate. As an advantage of this solution, it operates faster as less coordination is required.

These two options are easier to implement and manage in a short-term perspective. However, from a long-term perspective, they seem less robust than the one proposed below and are not aligned with the Aztec values.

There are also plenty of other options that didn’t come to my mind. However, I can’t compare them because they didn’t come to my mind.

Details

General Assumptions

- There is a forced exit mechanism;
- There exists a forum for discussing protocol/network changes (e.g., AZIPs);
- There is a reliable Soul Bound Token protocol that allows one to prove belonging to the organization without revealing identity.
- There are enough participants in the network to make the governance mechanism sustainable. Check the section “Questions” for further ideas on what is “enough” in this context.
- We prioritize decentralisation over speed.

Affiliated parties

[

Group 427318557 (2)

1920×1241 89.9 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/2e8634507058974334253485 4d8d0df24c6f76dc.jpeg)

Assumptions about parties

- Dapps are the heart of the ecosystem. That is what makes the chain unique. No dapps – no users. That is, dapps' opinion should have substantial weight, tho none of them should be able to have too huge influence (even if it has a large share of chain activity).
- By infrastructure providers, we mean RPC providers, oracles, indexers, interoperability layers, and other tools and pieces that help dapps to operate.
- Users (dapps users) should have an opportunity to express their opinions. However, it shouldn't be mandatory or play a core role in general mechanism sustainability.
- Dapps and infrastructure providers, if they wish to support the upgrade, are ready to take some actions, such as upgrading the software they run on a node, updating UI, etc.
- No specific actions are required from users for the upgrade.

Part 1: How do we agree that we want to upgrade to {X-release-of-software}

Disclaimer 1: Two types of upgrades take place in the rollup: (i) node software upgrade, that is, to upgrade node operators should upgrade the software they run, and (ii) smart contracts and bridge upgrades, which are executed on the chain side using multi-sig or other mechanisms. Below, we cover both cases, starting with "Smart contracts and bridge upgrades" and then modifying the scheme for node software upgrades.

Disclaimer 2: We assume that most proposals are reasonable and the proposed mechanism "Slow Soul" is aimed at their improvement, stimulating open discussion and collaborative work. However, in case of unreasonable or malicious proposals, the mechanism will provide a solution to decline the upgrade, empowering the community to decide.

Smart contracts and bridge upgrades

We consider "the vast majority" to be 90%+, as for the general chain health, it is reasonable to stay a single chain without breaking into smaller parts, as it affects a number of participants of the network while the last is closely affiliated with network security, censorship resistance, and liveness (e.g., if half of Decentralized Sequencer Operators stick to the old chain, and a half of them jump into a new version, the decentralization is decreased exactly twice).

However, we also want to provide a reasonable opportunity to fork the chain the community wants, so in the second iteration of the Review Stage in "Node software updates," we decrease this barrier to 70%.

[

1864×1186 88.5 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/988a859dd942e8f6e860e3b502aca2d199136d17.jpeg)

1. Initial Proposal:

An upgrade is proposed in full by the Aztec team on the community forum. The forum is available for anyone both to read and to comment. All interested parties (dapps and infrastructure providers) are notified about the proposed update.

1. Initial Review:

There is a 1-2 month period for everyone to review, comment, and offer suggestions. The specific period length is defined based on the update size. The larger the size, the more time is provided.

1. Community Call #1:

After the initial review stage is over, that is, everyone who was interested had enough time to review and comment on the proposal, Community Call #1

takes place. Community Call (i) covers questions discussed in the forum (discussion is continued, or if it is over in the forum, the solution is confirmed and shared with everyone), and (ii) opens the stage for other questions and suggestions that were not discussed before. Community Call is facilitated by an Aztec team member or a community volunteer.

1. Improved Proposal #1:

Based on the forum discussion and Community Call, the Initial Proposal is improved and published on the forum.

1. Voting #1:

7 days after the Improved Proposal was published, Voting #1

takes place. For the specific voting mechanism, check the next section, “Part 2: voting mechanism”. If the update is supported by 90%+, the upgrade is accepted.

1. Repeated Review #1:

If the update is supported by less than 90%, the process goes back to the forum for the Repeated Review. Its goal is to discuss the issues that the community disagrees with and find solutions to handle them. After one week of forum discussion, Community Call #2

takes place. Improved Proposal #2

is published on the Community Forum. Voting #2

takes place a week after Improved Proposal #2

was proposed. The whole Repeated Review stage takes nearly two weeks.

1. Repeated Review #2:

The mechanism for Voting #2

and its outcomes is exactly the same as for Voting #1

. If the protocol was supported by less than 90%, the Repeated Review can have one more round.

1. Rejected Upgrade:

However, if in Voting #3

, the upgrade was supported by less than 90%, the upgrade is rejected. A new version of the upgrade covering the same issues can be proposed not earlier than in 6 months.

1. Exit Opportunity:

Independently of the iteration at which the solution on the upgrade was made, the community should have not less than two weeks before the upgrade will be implemented to have enough time to bridge their assets back to L1 or anywhere else if they do not want to stay on the upgraded chain.

Node software updates

- Initial Proposal > Initial Review > Community Call > Improved Proposal:

Points 1 to 4 from the previous section should be repeated.

- Temperature Check Voting #1:

After the Improved Proposal, Voting takes place as well. However, its role is to conduct a “temperature check” and figure out if the community is aligned about the upcoming update or if more discussions and collaborative work are required.

- Repeated Review:

If less than 90% supported the upgrade, the Repeated Review takes place, followed up by the Temperature Check Voting #2

. The rules are exactly as in the previous section.

- Temperature Check Voting #2:

If more than 70% support the update, the new version of the software is released, and supporters upgrade it while those who disagree have an opportunity to fork the chain (more on forks in “Part 4: fork mechanisms”).

- Repeated Review:

If less than 70% supports the upgrade after the Temperature Check Voting #2

, the Repeated Review takes place again, repeating the stages of Forum discussion > Community Call > Improved Proposal > Temperature Check Voting. If less than 70% supports the update, the new software version is not released. A new version

of the upgrade covering the same issues can be proposed not earlier than in 6 months.

Note: We assume that most participants are aligned with the chain mechanism and philosophy and advocate for chain health at the current chain version. That is why the goal of the pre-voting stage of both upgrades described above is to help the community come to the majority alignment (if it is not there at the first Voting Round) or to reject the upgrade and allow to propose it again in 6 months as a substantially improved and reviewed version of the previous one according to community expectations and needs.

Part 2: Voting mechanism

- We propose using Soul Bound Tokens for Voting to provide the following demographics: 1 Vote = 1 Network Participant (Organisation). By Network Participant, we mean dapp or infrastructure provider. To make it real, we refer to the first General Assumption, saying that for this proposal, we assume we have a reliable SBT Protocol that (i) allows one to prove belonging to a specific organization without identity disclosure, (ii) allows us to conduct Voting without identity or organization disclosure with custom rules (such as 1 Vote = Organisation), (iii) and it is impossible to prove if one voted for or against the proposal (to prevent possible bribery attacks).

- The Soul Bound Token can be claimed in 2 months after the dapp was deployed on the network or the infra provider integrated the network. However, to participate in either Voting or Temperature Check Voting, the SBT should be claimed 6+ months ago. It softly prevents the voting mechanism from Sybil Attacks. Additional conditions on specific dapp characteristics to be legible to claim and SBT might be added to strengthen this mechanism.

- Dapps users can also have their votes counted, however, in a limited form: one user's vote counts for 0.01 SBT vote. To vote, the user can delegate their vote to the dapp. One user can delegate their vote only once and only to one dapp. The total number of votes dapp users can delegate to the dapp is limited. This limit depends on the total number of organizations that claimed SBTs: the total number of votes of each organization should be no larger than 5% of all claimed votes.

That is, for example, if there are 100 organizations in the network, each of them can claim one SBT, 100 votes is the total amount of all claimed votes. 5% = five votes. That is, if one vote belongs to the organization, four votes can get delegated from its users. 4 votes / 0.01 vote = 400 users can delegate their votes to each organization. This mechanism assumes that dapps users are softly counted. However, the dapp can't use their userbase to affect the Voting results substantially.

The user vote should be delegated for each proposal separately to prevent the organization from exploiting users' laziness or forgetfulness.

- The Aztec team doesn't have any SBT tokens and doesn't vote. Its goal is to facilitate the process and prepare and improve proposals based on the community feedback, needs, and expectations while staying neutral.

Part 3: How do we actually perform the upgrade to {X-release-of-software}

- For Node software updates

, each node operator is responsible for updating their software to a new version. The protocol used for a new software version delivery to each network participant should meet integrity security requirements. That is, the node operator can be sure that the new version of the software wasn't tampered or modified. The specific mechanism for software distribution security is out of the scope of this proposal, but as a raw example, some MAC standards (such as CMAC or HMAC) can be used as we care about data integrity (tampering possibility) without caring about data secrecy (data eavesdropping).

- For Smart contracts and bridge upgrades

, the multi-sig can be used. As a reliable multi-sig mechanism, the following is proposed: * High signers rate: 10/12 should sign the contract to upgrade it;

- For each update signing, 12 signers are chosen pseudorandomly from the pool of potential signers (there should be at least 36 potential signers in the pool, check the questions section for more thoughts about this number);

- Potential signers pool allows any organization with claimed SBT (dapp, infrastructure provider, and Aztec) to have one representative in the pool. That is, the principle of 1 organization = 1 representative is preserved;

- Using a reliable SBT protocol (General Assumption one), each representative in the pool proves that he is a unique representative of the organization without revealing their identity or organization. That is, both potential and actual signers don't know each other and can't coordinate.

- High signers rate: 10/12 should sign the contract to upgrade it;

- For each update signing, 12 signers are chosen pseudorandomly from the pool of potential signers (there should be at least 36 potential signers in the pool, check the questions section for more thoughts about this number);
- Potential signers pool allows any organization with claimed SBT (dapp, infrastructure provider, and Aztec) to have one representative in the pool. That is, the principle of 1 organization = 1 representative is preserved;
- Using a reliable SBT protocol (General Assumption one), each representative in the pool proves that he is a unique representative of the organization without revealing their identity or organization. That is, both potential and actual signers don't know each other and can't coordinate.

Part 4: fork mechanisms

By soft fork, we mean strictly reducing the valid set of transactions.

By strictly expanding hard fork, we mean strictly expanding the valid set of transactions.

By bilateral hard fork, we mean that the two rulesets are incompatible.

Starting with Node software updates:

- If an upgrade is a soft fork, nodes following the old rules will still get on the new chain. So, both forks can efficiently operate further without any barriers.
- If an upgrade is a strictly expanding hard fork, the old rules are a soft fork with respect to the new rules. Both forks can efficiently operate further without any barriers as well.
- If an upgrade is a bilateral hard fork, there should be enough participants to maintain the previous chain as, strictly speaking, it will be split into two separate chains. For Aztec, with decentralized sequencing/block building operators and provers (as intended), there should be enough fork participants to run each type of node software to provide the intended rate of decentralization.

However, in the case of a bilateral hard fork, the minority fork has to take care of all necessary infrastructure on their own. If there are not enough participants to decentralize the network enough, another mechanism can be chosen by participants on a temporal or permanent basis, such as the temporal usage of the centralized sequencer.

The core point is that nothing prevents the community from running previous node software and building the necessary infrastructure.

With Smart contracts and bridge upgrades, as contracts are upgraded either for everyone or for no one, the only option to maintain the previous version of the contracts is to fork the previous version and deploy it on their own, also taking responsibility for its maintenance. As the social consensus defines the "real" bridge or contract, as long as there are enough fork participants to define the fork as a "real" thing, there are no barriers for the community to fork any parts of the chain and use them further.

In the case of the Aztec bridge, the bridge fork is even easier as each dapp has its portal contract and can decide on whether to upgrade it (similar to how node operators make a solution on node software upgrade acceptance or rejection).

Questions

- What is "enough amount" in the network participants context as we assume using 10/12 multi-sig for upgrades execution? In fact, it means we require at least 12 organization representatives who self-volunteered to be in the signer pool. However, to make this multi-sig more reliable, we'd assume having at least 36 potential signers in the pool (the smallest number larger than 30 such that number % 12 = 0, where 30 is the minimal sample size to approximate a normal distribution acc. to the Central Limit Theorem).

Should we have a spare mechanism in case fewer than 36 representatives are in the pool? What mechanism should it be?

- What specific encryption mechanism should be used to distribute a new version of software to provide integrity? Are there any other security parameters that matter for this specific case?
- Can this design benefit from any time delay mechanism? From the first view, the suggested mechanism seems already slow enough to prevent the chain from any unexpected executions or actions. However, if working on this mechanism further, I'd watch at the edge cases of its components to check for any social vulnerabilities.