The next version of Aztec will be launched as a fully decentralized network. Community discussion and feedback plays a vital role in decentralization, and one of our goals is to build Aztec as transparently as possible. For key network decisions, similar to the Sequencer Selection RFP, we will be following a call for proposal method allowing anyone to submit proposals.

Up next on our roadmap: Upgrade mechanisms

Below you will find more information and learn how to get involved.

Overview

Sometimes it's necessary to release new versions of software. The ability for a community to carefully decide on and implement changes is key to a rollup's longevity and stability.

Aztec will have a fully permissionless set of sequencers, responsible for publishing rollups to the Ethereum mainnet. The network may also have a governance or staking token. All of these could be used to upgrade to a newer version of the protocol.

This request for proposals is asking the community to help us determine the best way to coordinate upgrades and how consensus on an upgrade occurs.

Specifically, it aims to address the following questions:

1. How do we agree that we want to upgrade to {X-release-of-software}
2. How do we actually perform the upgrade to {X-release-of-software}

Further context

For background on upgrades vs migrations and a framing of the problem at hand please visit the Appendix.

User stories

Proposals should try to cover all of the user flows below. Ultimately, some of these may be conflicting, as there are often tradeoffs in any circumstance which should be considered.

Users:

- What actions do users have to take in order to support, or reject, a network upgrade?
- When hard forks occur, individual users ideally should not have to change their wallets, keys, or necessarily do anything to support the latest version of the network.
- When hard forks occur, individual users should ideally have the option to opt-out of the changes, as well, via a forced exit, or another mechanism.
- When hard forks occur, individual users ideally should not have to change their wallets, keys, or necessarily do anything to support the latest version of the network.
- When hard forks occur, individual users should ideally have the option to opt-out of the changes, as well, via a forced exit, or another mechanism.

Developers:

- What actions do application developers have to take in order to support, or reject, different types of decisions?
- Ideally, developers have the option to use the "old" chain similar to using outdated versions of uniswap, or Ethereum classic.
- If wishing to support the latest version of the chain, application developers may need to update their UI &/or infrastructure to the latest version.
- Ideally, developers have the option to use the "old" chain similar to using outdated versions of uniswap, or Ethereum classic.
- If wishing to support the latest version of the chain, application developers may need to update their UI &/or infrastructure to the latest version.

Infrastructure providers:

- What actions do infrastructure providers have to take in order to support, or reject, upgrades?
- Ideally, infrastructure providers have the option to continue running the "old" chain in order to support any applications

still running on previous versions. It should only require a single, or a small number of sequencers running previous versions for them to still be practical.

- If wishing to support the latest version of the chain, infrastructure providers may need to update to the latest version.
- Ideally, infrastructure providers have the option to continue running the “old” chain in order to support any applications still running on previous versions. It should only require a single, or a small number of sequencers running previous versions for them to still be practical.

- If wishing to support the latest version of the chain, infrastructure providers may need to update to the latest version.

Requirements

Proposals must meet the following requirements and design considerations. If your proposal doesn’t meet all requirements, please call out the missing requirements:

Censorship Resistance:

- The proposed solution should be resistant to censorship and ensure that all nodes have equal access to information regarding the upgrades.
- Users must have the ability to exit and recover their assets back into L1 if they do not agree with the changes.

Grieving attacks or Malicious Upgrades

- The upgrade mechanism should not be susceptible to malicious upgrades.
- Similarly, the proposed mechanism should not allow a small minority to prevent an upgrade.

Well-Known Mechanism:

- The upgrade mechanism should be easily understood and “well known” in advance by the entire network, to foster trust and transparency, allowing users to make educated decisions.

Public decision making:

- The mechanism should use some form of social consensus or community decision making to ensure that the community is in support of the upgrades, and that no single entity has control over the decision-making process, nor victim to potential bribery attacks.
- This does not mean it cannot use some other mechanisms beyond social consensus, or combination of mechanisms, such as token weighted voting, or otherwise. It is defined as a requirement to ensure that whatever mechanism is used accurately reflects the community’s sentiment, rather than other types of ecosystem actors.

- This does not mean it cannot use some other mechanisms beyond social consensus, or combination of mechanisms, such as token weighted voting, or otherwise. It is defined as a requirement to ensure that whatever mechanism is used accurately reflects the community’s sentiment, rather than other types of ecosystem actors.

Assumptions

1. There exists a mechanism to force exit (withdraw assets from a L1 portal)
2. There exists a mechanism to force inclusion (push tx’s into the L2)
3. There exists a forum for discussing protocol/network changes (e.g. AZIPs)
4. A network token can optionally exist.
5. One of the submitted sequencer selection proposals will be chosen for how the sequencer participates in the network, and therefore, participates in upgrades
6. Due to the flexibile programmability in Aztec, there is no canonical bridge and each application on Aztec may have it’s own “portal” contract for interacting with L1. Portal contracts hold assets independently. The Aztec rollup contract consists of a state transitioner and “message box” contracts. Neiter of these hold assets. Read the docs for more information about Portal Contracts.

Submission Format

To ensure consistency and facilitate the review process, kindly adhere to the following submission format:

- Title: A concise, descriptive title for your proposal

- Summary: A brief, easy to understand summary of your proposal (about 300 words)
- Comparisons: Explain what makes this solution unique and different from alternative solutions
- Details: Explain the upgrade mechanism, its components, and its functionality
- Questions: Any outstanding questions

Submissions should be created as a new post on this forum, tagged upgrades

and rfp

. Once the new post is created, please refer back to this RFP and post the link to your proposal as a comment.

Grants

Complete proposals may be eligible for a retro-active cash grant and swag.

FAQ

We anticipate that you may have questions regarding the call for proposals. The following frequently asked questions and their corresponding answers should provide some clarification. Otherwise, feel free to post a question in the forum, contact cooper@aztecprotocol.com, or follow us on Twitter for updates.

Q1. How will a proposal be chosen?

A1. Proposals will be evaluated based on their adherence to the requirements and design considerations, as well as the quality, feasibility, and innovation of the proposed solution. The selection committee, consisting of Aztec Labs employees and possibly external stakeholders, will determine the winning proposal and share the chosen solution publicly.

Q2. Who can submit proposals?

A2. Anyone!

Q3. Can I submit more than one proposal?

A3. Yes, you can submit multiple proposals if you have different ideas for upgrade mechanisms.

Q4. What if my proposal does not fully meet the requirements?

A4. We still encourage you to submit your proposal and participate in the discussion, as your ideas could contribute valuable insights and help shape the final solution.

Appendix

This is a tough problem & you probably need some additional context!

Here's a nice overview from@LasseAztec

Difference on migrations and upgrading

Generally, we make a distinction between a "technical upgrade" and "technical migration". A migration is a separate deployment, where users must take action to migrate their positions from A

to B

. An upgrade instead requires no interaction for the individual user.

Example of migration: Uniswap

: With the launch of V3, people would migrate funds from V2 to V3 to supply liquidity.

Example of upgrade: Aave

: The pool contract is using a proxy and the implementation have been updated on occasion.

While the end goal for those are similar, "provide users access to a new version of the software", it have some major differences it how this is acheived, and what kind of actions is required by the users of the software. This distinction is likely to be relevant to all of the proposals.

Ethereum

In the domain of Ethereum, deciding that we want to upgrade, and what to upgrade to is often a mix of discussion by the community proposing EIP's and client developers that work on implementing the proposed EIP's. When a decision is made to include XYZ into a network upgrade, a time of upgrade is decided. Throughout this process any community member can raise issues with the proposed changes and you might see fractions in the community that disagree whether to support or reject the change.

When the time of the upgrade arrives any node in the network can decide if they want to run the new node software (including the change) or continue running the existing software. If there is a significant divide between the nodes this can spawn long-lived forks of the network with different rules. The value of the fork (and its claim to be the real

Ethereum) comes down to social consensus. A famous example of such a split spawned Ethereum Classic after the Dao Hack. Social consensus decided that classic was not the real

Ethereum, but both versions live on and users are able to decide for themselves what they think is the real

Ethereum.

Rollups with bridged assets

As Jon Charbonneau

nicely conveyed in his essays Rollups Are L1s (& L2s) a.k.a. How Rollups *Actually Actually Actually* Work and The Rollup Multiverse a rollup can use the same structure as Ethereum to figure out when to fork and what to fork to but there is one very large difference from a fully separate blockchain - the bridged assets

(foreign state).

If there is a proposed upgrade and community want to "fork off" they can deploy the rollup state transition contract, and start it from the current state. The assets can be used inside the fork as if they always were there, but the collateral for any bridged assets will still be in the same contract (not forked by this).

The bridge must decide which of the forks is the "real"

one, and only allow it to withdraw funds from the bridge. Only one of the forks can be seen as the "real"

one as it might be possible to double-spend otherwise. As an example, assume that you have 10 eth and the fork happens. On both of the forks you now have 10 eth, so you can from the L2 point of view exit those to L1. If L1 accepts messages from both forks you would be able to exit with 2*10 eth and practically have stolen someone else funds (double-spending you deposit).

If the contract with the collateral is controlled by some governance that decided what it seen at the real fork it can impact what is seen as the "real" fork by anyone. For the case where a rollup has a canonical bridge which holds custody of a large part of the rollup value it can practically decide what is seen as the real fork (or at least which of the forks that will have backed assets).

[

"Rollup from L1 POV

1920×2276 71.1 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/6c031e6e8cc5bfa3292363e89a6889ee201fbd3a.jpeg)

Above is a diagram showing how such a setup could look. We are showing message boxes as the means of communicating between L1 and L2. This set up itself is a consideration that may be relevant to your proposal! A simple (perhaps naive) proposal could define a multisig across each of these L1 contracts which can upgrade as they see fit. As you can imagine, there is a large design space.

Governance Spectrum

Generally, all proposals will live on a spectrum from "strong, well defined governance" to "weak" or almost non-existent governance (e.g. everything is a non-upgradable contract).

[

Ungovernable

3368×753 78.3 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/ab85271e07a15518a2464506b301782993931d59.jpeg)

Ungovernable

If the contracts on L1 seen in the above diagram are all immutable without admin control or keys, the act of upgrading from one version to another requires that users are exiting funds from the rollup to enter in another system. This is sometimes referred to as a migration

as users need to migrate

the state from one system to another to employ the benefits of the new.

It is a “upgrade” mechanism that have been used in multiple projects in DeFi mostly prominent is probably Uniswap V3, which required liquidity providers to exit V2 and put their funds into V3.

Governed

Alternatively, contracts can be controlled by some governance organ which are able to decide what is seen as the “real” fork.

The exact mechanism for how this decision is enforced varies between different platforms, but can be done by having a proxy in fron of every contract such that the implementations can be changed by govenrance if the decision is made. Graphically this would look like what we saw earlier, just with an extra party the “governance” that can update the implementations.

[

Swiggity

1920×1910 70.8 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/bfcf2988381024544e0262fe46fe957e9a0665e4.jpeg)

Upgrade delay and rage-quitting

To ensure that it is possible to rage-quit the system in case an undesired upgrade, it must be possible for the user to perform a forced-exit before the upgrade is executed. Otherwise malicious governance can trivially steal all bridged funds as we have recently seen in other projects. This is likely a common feature that will be seen across many proposals and authors can assume this mechanism exists.

Tradeoffs

To outline some of the tradeoffs in a digestable manner, the below table shows what is required by the user to ACCEPT or REJECT upgrades and outlining what the user trust beyond correct implementation.

[

Governance variations and their properties around upgrades

1920×2578 215 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/364d7eb432b0952e2c1f21f261fca18a51e52989.jpeg)

To summarise:

- Upgrade mechanisms that use governance do their job for delegating decision making which makes it convenient for the end-user as long as they agree with the governance direction.
- Upgrade mechanisms that are fundamentally immutable (ungoverned) prove to be a double-edged sword that can protect us from others but may harm us if we are not ready for it.
- The goal of this RFP process is to identify solutions that fit on this spectrum, and what the right tradeoffs are for Aztec! While it sadly isn’t this binary in reality, we expect all proposals to fit somewhere in between, or leveraging combinations of this identified design space.

DISCLAIMER