Aztec is allocating up to $100,000 as stipend to facilitate building the first privacy focussed “Etherscan” of the crypto ecosystem

We are calling all independent builders and projects to submit their proposals to build an Aztec block explorer. We will select up to3 proposals and support teams through the testnet launch via milestone-based stipends. We will only select projects building for the long-term (beyond testnet!). If interested please keep reading to find out how to apply.

Proposals must be submitted before 22 June 2024 to be considered for the stipend.

Overview

Why do this? Aztec is launching as the first decentralized L2 and a key pillar of decentralisation is that core infrastructure is built by the community. Secondly, our goal is to build Aztec as transparently as possible via open-source product development. As such all decisions are posted publicly for community feedback and participation. Please see previous RFP’s on Sequencer Selection and Upgrades) for examples.

The first of these infrastructure projects is: Block Explorers

Traditionally, block explorers enhance trust and accountability within the blockchain ecosystem by allowing people to monitor the state of the network and transaction activities. For privacy-focused networks like Aztec, where one cannot monitor user activity or learn much about a transaction, a block explorer may appear limited but is still critical for users to view and verify transactions and monitor the health of the blockchain.

As Aztec prepares for testnet launch, a usable block explorer is a P0 for users and developers building applications. Check out Helpful Reference section and Appendix 1 for more info.

If you are not interested in exploring block explorers (pun intended), please refer to Appendix 2 on Potential Areas of Interest

Requirements

- Comprehensive and up-to-date data coverage of all blocks and events: Stay up to date with the Aztec blockchain providing complete information on all blocks, transactions and contracts. See Appendix 1 for a sample list.
- Search functionality: Users should be able to search for specific transaction hashes, block numbers or contract addresses. Ref Appendix 1.
- User-friendly interface: Should be intuitive and easy to use to search/view data easily.
- Responsive design: Should also be functional across various devices and screen sizes, including mobile phones and tablets.
- Reliability and uptime: The explorer should have high availability and minimal downtime to ensure that users can access the data whenever needed.
- Privacy considerations: Do not track user IP addresses to maintain a list of what a user searched.

Grant Details and Submission Format

Aztec’s execution environment is rapidly under development. To reduce developer friction, teams should start building in August and aim to finish building by November. We are willing to give a stipend to support independent developers or small companies with the overhead of onboarding to Azte,. We are looking to support up to 3 such teams.

We will bias towards teams that have a clear vision of building for Aztec mainnet and beyond and support the developer ecosystem!

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

Title

: Team/Product name

Contact Details

: where Aztec Labs can contact you - telegram, email etc.

Summary

: A brief, easy to understand summary of your proposal (about 300 words)

Estimated Start and End Date

: recommended to start in August and end by November.

Details

: Any designs (UIs or technical system designs), features to support, details about your team and their expertise or relevant experience etc. Make this section unique and try to stand out from other proposals.

Grant Milestones and roadmap

: 3-5 milestones on how you expect to get to a testnet ready explorer, with at least one item dedicated to future work (post testnet improvements, feature additions, and mainnet launch). Importantly please indicate your longer term roadmap and how you o want to keep building in the long run as an independent company.

Grant amount requested

: Any grant requests over $45,000 will be immediately rejected. Successful projects may be eligible for retroactive grants at a later time.

Grant budget rationale

: explain comprehensively, why you are requesting the amount of funds

Questions

: Any outstanding questions

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

and rfgp

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

Please submit proposals by 22 June 2024.

Helpful References

There is a lot to know about Aztec that informs the requirements and the design of a block explorer. We therefore highly recommend playing with the Aztec Sandbox or speedrunning through our tutorials before proposing a grant.

Aztec supports both private and public transactions. transactions start off private and private functions can enqueue public function calls. Users create a proof of the private transactions client side and via a mempool send the tx hash, note hashes, nullifiers, encrypted (and unencrypted) logs to the sequencer. The sequencer then executes any remaining public functions, creates the final tx proof and builds a L2 Block. Notice that the block only hasstate diffs (aka transaction effects) and not calldata of individual transactions. This is because for private transactions, the sequencer doesn't know what the user executed. So a block explorer on Aztec will not be able to show transaction history of a user or full details of a transaction. Yet it is critical for users to know if the transaction is confirmed in a UI friendly way.

Block explorers also show a lot of fee related data. Aztec takes a unique position on fees.

1. Dapps can pay fees for users if they have an entrypoint

method.

1. Users (or dApps) can pay fees using the native fee paying asset (FPA) (e.g. ETH on Ethereum or STRK on Starknet) or via a "fee paying contract" that that can accept custom tokens based on its code or via L1->L2 message.
2. Fees can be paid either publicly or privately and refunds can be received in both private and public domains
3. Ultimately, the protocol only accepts the FPA. How much of the native FPA was paid for a given transaction is public knowledge.

Based on what information is public, a block explorer should show appropriate fee data.

Block Explorers also show data related to a smart contract.Aztec has native account abstraction, which means there are no EOAs. Every address is a contract. Additionally, taking inspiration from Starknet and object oriented programming, Aztec has contract classes and instances. Contracts in Aztec are deployed as instances of a contract class. Deploying a new contract then requires first registering the class, if it has not been registered before, and then creating an instance that references the class. This means if the same contract is being deployed, it is much cheaper as you just have to instantiate the class. Contracts have public bytecode (for public functions) and private bytecode (for private functions). For the latter only a commitment of the code is deployed on chain as opposed to the entire code. Yet block explorers can play a role in verifying both the bytecodes.

Aztec has 3 kinds of logs:

- unencrypted logs - these are public.
- encrypted logs - these are private, and are encrypted to who the receiver of the log would be
- encrypted note logs - these contain the encrypted content of a note that the receiver needs to build their notes.

Logs may have "tags'' associated with them to enable a user to discover their notes from our notes hash tree. Without such tags, a user may have to brute force the entire note hash tree to figure out which note hashes are theirs. We are still implementing and finalizing our note discovery scheme but we will have a registry where an account will mention what discovery scheme they use (e.g. brute force, note tagging etc.). A good place to read up on this is here and our protocol specs

Finally, a note (pun unintended) on Keys on Aztec - unlike other blockchains which just have a single private-public key pair, Aztec has several different keys to separate concerns. One such key is the “incoming viewing key” which can decrypt notes that were sent to the user. This way, the user can use this key to find out note contents, decrypt encrypted logs to them etc. You can find more information here. Similarly, we have an “outgoing viewing key” for a user to decrypt any log/data for when they created a transaction for another party. A fully functional block explorer may allow users to view a personalized dashboard of their transactions (e.g. on client side, the viewing key could decrypt all transaction logs and note hashes and reveal to the user what these contain).

FAQ

If you have any questions feel free to ask them here or reach out to rahul[at]aztecprotocol[dot]com

Appendix 1

What should a user see on Aztec?

For a given proved block number, a user should be able to see:

- block hash
- timestamp
- number of transactions and their transaction hashes
- stats on logs
- block header data
- User friendly way of viewing transaction state effects
- Any gas related information (e.g. total fees, fee used up for L2 gas, Data availability, L1 gas etc.)

For a given transaction, a user should be able to see:

- Transaction hash (identifier)
- status (mining/preconf/proving/success/dropped)
- block number
- L1 batch transaction (Rollups tend to post and proves a batch of L2 blocks in one L1 transaction. Aztec may do this too, although currently, Aztec posts a L1 transaction per Aztec block with its proof)
- timestamp
- transaction fee (in the native fee paying asset)
- Transaction effects
- num notes + note hashes and note encrypted logs
- num nullifiers + hashes
- l1->l2 message hash
- l2->l1 message hash

- encrypted logs
- unencrypted logs
- public state reads/writes
- public function call(s) [similar to how you can view function calls on etherscan]
- num notes + note hashes and note encrypted logs
- num nullifiers + hashes
- l1->l2 message hash
- l2->l1 message hash
- encrypted logs
- unencrypted logs
- public state reads/writes
- public function call(s) [similar to how you can view function calls on etherscan]
- Fee related information
- FPA used
- gas price
- gas
- fee divided per dimensions (DA, L1, L2)
- coinbase (recipient of block reward)
- fee recipient (Address to receive fees)
- any public teardown call
- any public fee information (depends if the user/dapp pays fee in public or not):
- fee payer/fee payment contract or if paid from L1.
- tokens paid (if paid in a custom token)
- intial fee bid in token or FPA
- public refund received
- fee payer/fee payment contract or if paid from L1.
- tokens paid (if paid in a custom token)
- intial fee bid in token or FPA
- public refund received
- FPA used
- gas price
- gas
- fee divided per dimensions (DA, L1, L2)
- coinbase (recipient of block reward)
- fee recipient (Address to receive fees)
- any public teardown call
- any public fee information (depends if the user/dapp pays fee in public or not):

- fee payer/fee payment contract or if paid from L1.
- tokens paid (if paid in a custom token)
- intial fee bid in token or FPA
- public refund received

- fee payer/fee payment contract or if paid from L1.
- tokens paid (if paid in a custom token)
- intial fee bid in token or FPA
- public refund received

Note there is no from

or to

because transactions start private first. To

can optionally be the first public contract call that isn’t a fee paying contract.

For a given (contract) address, a user should be able to see:

- contract public code (or bytecode)
- contract private code (or bytecode)
- contract class ID
- version
- hash of the initialization function that ran
- deployer address
- salt
- note tagging scheme registered
- public keys assosciated to the address (e.g nullifier, viewing etc)
- hash of public keys deployed with the contract
- Any public transactions where this address was either a from

or a to

Note this list doesn’t contain anything private that a user may be able to decrypt using their viewing keys

Appendix 2 - Other Potential Areas of Interest

This section explores other things tangential to a block explorer, if you are still reading this document and don’t want to build an explorer itself but think about the peripheries around it. In order of importance to Aztec Network:

Viewing key tool

- this block explorer won’t ever show you your own transaction history. Aztec has a notion of viewing key that let’s you decrypt any logs or notes that belong to you. Using the viewing key, you can decrypt transactions to learn about your own transaction history. This tool can be used by explorers, wallets or even apps as a browser extension. Viewking keys can be derived on a per app basis. So an app can use this tool to show you your past transactions on the app. Obviously this decryption has to happen client side otherwise you would leak your viewing key to the app/tool.

Programmatic access to node data

- some explorers now offer API access to query node data. This can be offered at the explorer level but also at the node level. An app, wallet or explorer can later integrate this.

Indexers

- Same as other blockchain networks.

RPC providers

- Same as above

Bootstrap nodes

- these are special kinds of nodes designed to let your node sync quickly to the blockchain. This can be quite useful for anyone running their own node like a user, wallet provider, explorer, etc.

If any of these interest you, please reach out to rahul[at]aztecprotocol[dot]com! I look forward to speaking to you.

DISCLAIMER