Zenith Full Node Wallet

Pitch

The Zcash ecosystem needs a modern full node wallet and Zecwallet has left a gap that we intend to address with this grant.

Applicant Background

We are a father-and-son team with decades of combined experience in technology consulting, project management, digital product management in South America and the United States.

- Rene Vergara Larrea, MSc, PMP
  - Senior Technical Product Owner for a Fortune 50 insurance firm in the US, working with Java, ReactJS and MongoDB.
  - Creator of ZGo.
- Rene Vergara Araque, MBA, MPM
  - Developer for ZGo, maintainer of ZGo's website.
  - Creator of GeoVSU, a GIS-enabled web application for the real estate market in Ecuador.
  - Creator of the educational videos for GeoVSU.
  - Former CEO of AT&S Consultores, a technology consulting firm in Quito, Ecuador.

Opportunity

The first product derived from our work in building ZGo is Zenith, the Zcash Full Node Command Line Interface (CLI) wallet. Zecwallet has left a void in the Zcash ecosystem for users that run their own full nodes and use the node's wallet for ZEC transfers. Zenith can fill this void.

Also, with the announcement from ECC that `zcashd` will be retired on 2024, Zenith can provide the full node wallet functionality on `zebra`.

Building a modern full node wallet allows us to make many improvements that have been discussed by the community:

- Deprecate standalone transparent addresses, only supporting them under Unified Addresses.
- Remove transparent-only versions of the RPC endpoints.
- Implement Unified Viewing Keys with functionality already available in `zcash-haskell`.
- Implement auto-shielding of ZEC and auto-migration to the Orchard shielded pool.

Proposed Solution

Create a full node wallet with a Graphical User Interface (GUI) that allows users to send and receive Zcash transactions and perform other tasks related to full nodes. The Zenith Full Node Wallet shall be built using the same full node RPC endpoints that `lightwalletd` uses. This allows our team to create a wallet that can work with `zcashd` or `zebra` as the full node implementation.

Solution Format

The new Zenith Full Node Wallet shall have a GUI with functionality to:

- Create new accounts and new addresses.
- Display received transactions.
- Send ZEC to Unified, Sapling, and transparent addresses.
- Send shielded memos to Unified and Sapling addresses.
- Maintain a local address book.
- Generate Payment Request URIs and QR codes[1].
- Accept Payment Request URIs to generate transactions.
- Load Sapling and Unified viewing keys.

The new Zenith Full Node Wallet shall have a set of RPC endpoints that allow applications to interact with the wallet programatically via HTTP requests.

The new Zenith Full Node Wallet shall have a CLI to provide full nodes without graphic interfaces (cloud servers) with wallet functionality.

Technical Approach

Figure 1 shows the proposed architecture for Zenith. The GUI shall be built on OpenGL and Haskell. The wallet shall leverage the functionality available on the `zcash-haskell` library and new enhancements planned to interact with `zcashd` or `zebra` and provide additional functionality. The transaction storage shall be built on a local SQLite database for portability and ease of installation. The wallet RPC server shall be built on Haskell and run on a different port than the RPC server provided by `zcashd`/`zebra`.

Figure 2 shows an example of the graphical capabilities of this approach.

Dependencies

- `zcash-haskell` library
- `zcashd` functionality
- `zebra` support for the RPC endpoints used by `lightwalletd`:

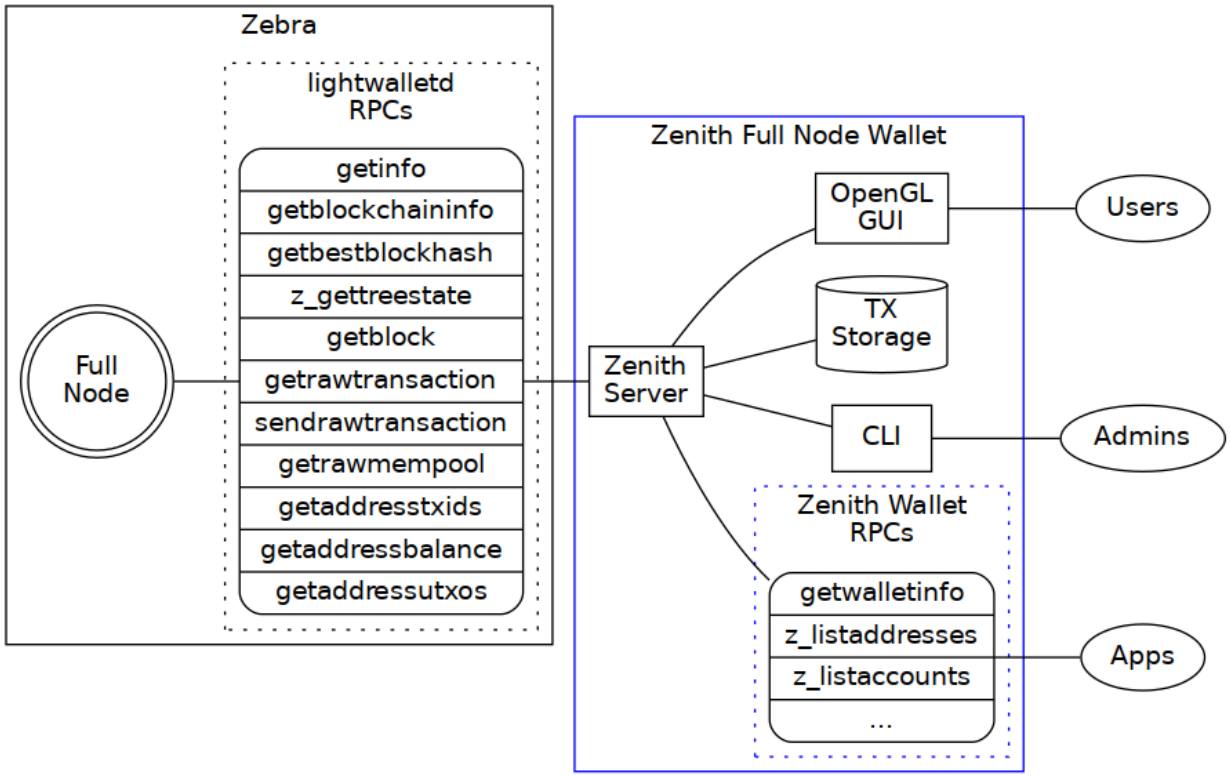

Figure 1: High level architecture of Zenith

- `getinfo`
- `getblockchaininfo`
- `getbestblockhash`
- `z_gettreestate`
- `getblock`
- `getrawtransaction`
- `sendrawtransaction`
- `getrawmempool`
- `getaddresstxids`
- `getaddressbalance`
- `getaddressutxos`

Execution Risks

Zenith is an application that relies on a Zcash full node for its functionality, therefore changes to the full node implementations can potentially impact Zenith's features.

Our proposed solution assumes that `zebra` will have all the RPC endpoints needed for our application implemented in a similar way as they are implemented by `zcashd`. Any large divergence in functionality

Figure 2: GUI proof-of-concept using Haskell and OpenGL

between `zebra` and `zcashd` will likely require additional work for Zenith to support `zebra`.

To mitigate this risk, our proposed solution uses a small subset of the RPC endpoints that are common between `zcashd` and `zebra`.

Unintended Consequences

Having the functionality of the wallet separated from the full node itself may mean that apps that interact with `zcashd` full nodes will have to update their systems to use Zenith. Also, there may be some duplication of effort with other teams in the ecosystem.

Budget

The estimated total cost for this proposal is $124,800.00. This estimate includes the cost of 39 weeks of feature development, estimated at 40 hours per week at a rate of $80 per hour.

Schedule

Based on our estimate of 39 weeks of work for the team, we propose a delivery schedule with four milestones as shown on figure 3.

We have three main components, the Command Line Interface, the Graphical User Interface, and the Applica-

tion Programming Interface. In order to release working products on a regular cadence, we have separated the functionality of the wallet into:

- Basic: features to create a new wallet, new addresses, send and receive Zcash transactions.
- Address Book: features to maintain a list of contacts with corresponding Zcash addresses.
- URIs: features to accept and generate payment URIs that are compliant with *ZIP-321: Payment Request URIs* [1].
- Viewing Keys: features to generate viewing keys.

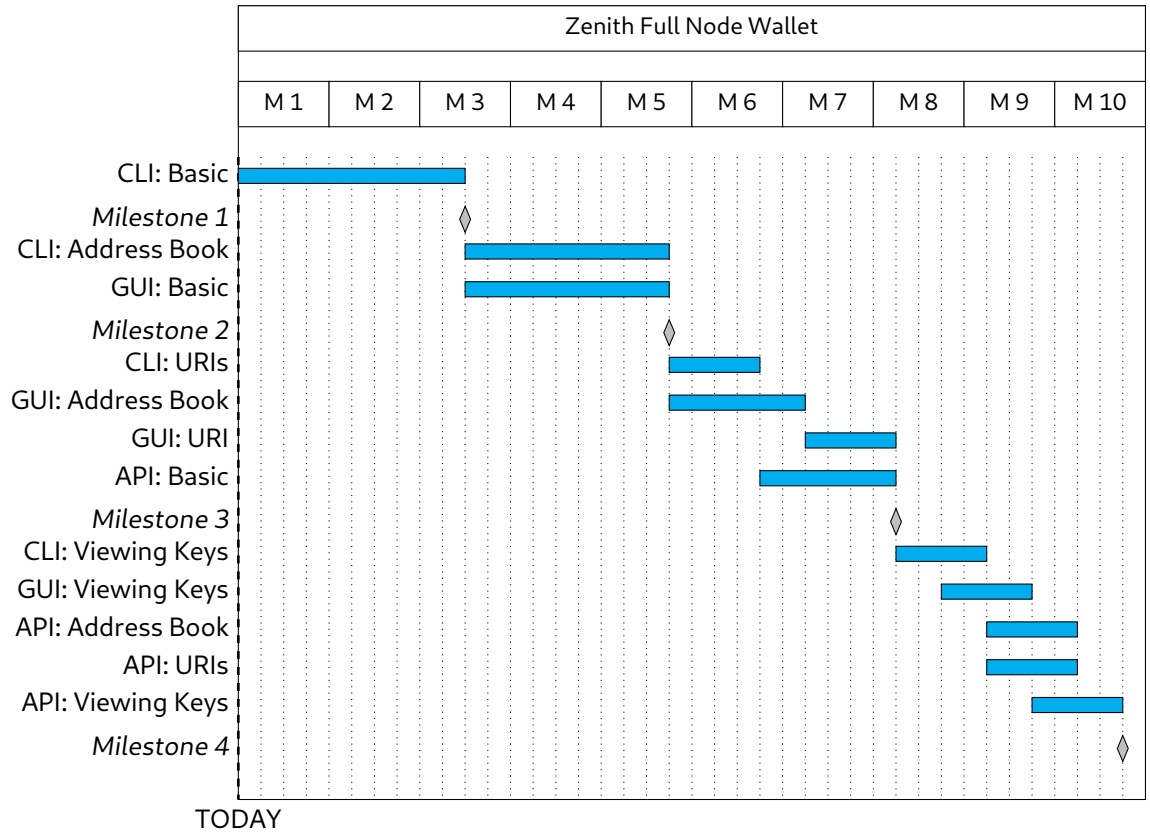

Figure 3: Schedule

References

[1] *ZIP-321: Payment Request URIs*. URL: https://zips.z.cash/zip-0321.

Acronyms

API Application Programming Interface. 4

CLI Command Line Interface. 1, 2, 4

GUI Graphical User Interface. 2, 4

RPC Remote Procedure Call. 2–4