Papers

Beacon Kusama - Report

Report 1
May 2021 – September 2021

This report outlines the effort of Papers to fulfill the requirements of the Kusama Bounty "#3 Mobile Signers / Injectors / Walletconnect equivalent"[1].

Important note
Due to the uncertainties around the distribution of the bounty, as it was initially planned to be split up into multiple sub bounties, Papers had to temporarily halt the development in September until a viable solution was reached. The development will be picked up as soon as a first installment was made, at least partially covering the development cost accrued until now.

1. Intro

During the requirements engineering phase changes it became apparent that there will have to be a scope change from the initial proposal.

This report outlines the deliverables for the **Phases 0 to 2** of this adapted roadmap.

2. Tasks

The following tasks were completed during the development phases under this bounty.

2.1. Beacon Kusama

Beacon Kusama focuses on enabling wallet and decentralized application developers to build on the Beacon functionalities to simpler interact between wallets and dApps.

Requirements engineering
During the requirements engineering phase, the development work was structured, and the assumptions made in the initial proposal were validated. These discussions were made with players in the ecosystem, namely the Fearless Wallet team and Pocket 4D.

One main takeaway was that in the initial proposal the mobile native Beacon SDKs for Swift and Kotlin were not in-scope. As many wallets utilize these languages to build their mobile first wallets, the decision was made to change out the browser extensions (Spire) related parts with the integration of the Swift and Kotlin SDKs. The work on these native SDKs focuses on the WalletClient implementation.

Further integration work can be included at a later stage in addition to the work covered by the initial proposal.

The document with the findings is attached as **Appendix 1**.

[1] https://kusama.polkassembly.io/bounty/3

Polkadot Standards Proposal – PSP[2]

Based on the finding during the requirement engineering phase and the previous work Papers has already conducted with Beacon, a Polkadot Standards Proposal was drafted. The proposal has been shared with the community and is currently awaiting feedback from other parties.

Beacon SDK (Typescript)

The Beacon SDK implements the following:

- WalletClient – for wallet developers
- dAppClient – for dApp developers

The Beacon SDK for Kusama integrates a proof of concept in conjunction with the polkadot.js library. Polkadot.js that connects to an RPC node and prepare the transactions according to the metadata/runtime of the selected chain. The Beacon SDK provides a "BeaconSigner" that can be passed as a signer to the polkadot.js injector.

The specific messages that can be exchanged between dApps and wallets will be defined gradually. A simple permission request/response is defined, as well as a message to sign a payload.

Additional work has been done in splitting up the Beacon SDK into multiple packages, to allow developers to use only the necessary dependencies.

- Beacon SDK with multi package implementation[3]

Beacon example dApp[4]

The Beacon example dApp acts as a Beacon reference implementation for dApp developers. It is currently used in conjunction with the Fearless Wallet Beacon proof of concept.

Video[5] Fearless Wallet integration of the Beacon Kusama proof of concept.

Beacon Swift and Kotlin SDKs

The following two repositories are relevant for the mobile native Beacon SDKs.

- Beacon Kotlin SDK[6]
- Beacon iOS SDK[7]

The Beacon iOS and Kotlin SDKs handle the WalletClient equivalent of the Beacon SDK (Typescript). Besides the development conducted for this and in preparation to handle multiple protocols, these SDKs were split up in multiple modules. Multiple modules enable the developers to select only the modules relevant for them without having to bloat application with unnecessary dependencies.

The project consists of the following modules:

[2] https://github.com/w3f/PSPs/pull/29
[3] https://github.com/airgap-it/beacon-sdk/pull/253
[4] https://github.com/airgap-it/beacon-polkadot-example-dapp
[5] https://drive.google.com/file/d/1GQuQzbOiPOPdd5N7gkZqw3L6cs2pKfBc/view
[6] https://github.com/airgap-it/beacon-android-sdk
[7] https://github.com/airgap-it/beacon-ios-sdk

- core - common and base code for other modules
- client-wallet - the wallet implementation of Beacon
- client-wallet-compat - a supplementary interface for client-wallet for use without Coroutines
- blockchain-PROTOCOL - a set of messages, utility functions and other components specific for PROTOCOL
- transport-p2p-matrix - Beacon P2P implementation which uses Matrix network for the communication
- demo - an example application

Beacon Node infrastructure setup

As Beacon mainly relies on a matrix[8] based infrastructure, which allow a decentralized approach at running these nodes, an initial testnet was created with two nodes. Request related to the Kusama integration in this phase will be relayed through these nodes until more nodes will be added when the integration becomes production ready.

- beacon-test-node-1.crystal.papers.tech
- beacon-test-node-2.crystal.papers.tech

3. Spent resources

The following outlines the resources used for the following tasks.

Task	**Days**	**Dollar value**
Requirement engineering	15 days	18’000
Polkadot Standards Proposal – PSP	8 days	9’600
Beacon SDK (Typescript)	34.25 days	41’100
Beacon example dApp	11.5 days	13’800
Beacon Swift and Kotlin SDKs	31.5 days	37’800
Beacon Node infrastructure setup	5 days	6’000
Total	**105.25 days**	**126’300**

4. Next steps

- Completion of multi modules implementation
  - Beacon SDK (Typescript)
  - Beacon iOS SDK
  - Beacon Android SDK
- Definition of Substrate (Kusama/Polkadot) related messages
- Implementation of Substrate (Kusama/Polkadot) related messages
- Beacon Kusama integration in AirGap
- Developer documentation

5. Appendix

- Appendix 1 – Requirement engineering documentation

[8] https://matrix.org/

Papers

Beacon SDK – Requirement engineering documentation

Version	Date	Author	Changes
v1.0	30.05.21	Andreas Gassmann	Initial document
v.2.0	24.06.21	Pascal Brun, Andreas Gassmann	Changes and review

1. What is Beacon?

When interacting with a decentralized application, users want an approach that is as seamless as possible by pairing their favorite wallet effortlessly with the application.

With Beacon, a user can pair a mobile wallet to a decentralized application by scanning a QR code and establishing an encrypted channel over the decentralized Beacon peer to peer network. But Beacon also provides support for browser extension, web, desktop and hardware wallets that can build on the Beacon standard, allowing users dApp interactions with their flavor of wallet.

Thanks to Beacon developers don't need to implement yet another wallet solution for their application impacting the user experience for the end-user as well as requiring additional resources from the developer which would be better invested in the core features of their application.

2. How it works

Beacon builds on a decentralized peer-to-peer network built on the matrix protocol to act as a transport layer for the messages sent between the wallet and the application. Contrary to other implementations like WalletConnect where a centralized relay server is being used, Beacon brings additional security through decentralization.

A reference node has been created and multiple nodes have already been set up by Papers as well as other parties. This network can be discrectionaly extended by entities hosting their own nodes.

2.1. Goal

The goal of Beacon is to define a wallet interaction standard that can be used across blockchains and provide a seamless experience across devices and wallets over a decentralised peer-to-peer network.

2.2. Approach

Currently, the Beacon SDK is heavily tied to Tezos. While the general functionality is blockchain agnostic, some of the inner workings are tied to specifics of the messages that are being sent and received. As part of the KSM addition, we want to change the architecture and make it completely blockchain agnostic.

Another shortcoming of the current version of the Beacon SDK is that all the code is released in a single package. We will address this and split it up into multiple packages.

In order to allow other teams to start working with the Beacon SDK as soon as possible, we will start by providing a Beacon SDK version that can be used with KSM to share accounts and sign operations. This SDK version will be released for testing purposes only to gather feedback and help other teams getting started as soon as possible by having a working Proof of Concept available.

After this is done, we will start with the re-architecture of the Beacon SDK. The expected output is the following:

- The Beacon SDK will be split up into multiple packages. As an example:
  - @beacon-sdk/types
  - @beacon-sdk/core
  - @beacon-sdk/dapp-client
  - @beacon-sdk/dapp-ui
  - @beacon-sdk/wallet-client
  - @beacon-sdk/ksm
  - @beacon-sdk/dot
  - @beacon-sdk/xtz

The protocol specific packages will contain definitions and logic for that protocol.

The Beacon SDK will be used in conjunction with the polkadot.js library. Polkadot.js will connect to an RPC node and prepare the transactions according to the metadata/runtime of the selected chain. The Beacon SDK will provide a “BeaconSigner” that can be passed as a signer to the polkadot.js injector.

The specific messages that can be exchanged between dApps and wallets will be defined gradually. In the beginning, a simple permission request/response is defined, as well as a message to sign a payload. As time goes on and more use-cases will become clear, those and additional messages will be created and refined. Things to keep in mind are Encryption/Decryption and Push Notifications.

3. Native SDKs vs Spire

Spire, the browser extension initially developed alongside the Typescript version of the Beacon SDK enables communication between dApps and wallets outside of the Beacon Node network and works directly with post message.

During the discussions with team in the ecosystem, it became quickly apparent that the requirements in terms of integration would also need to focus on SDKs besides Typescript namely for Swift and Kotlin. Thus, the decision was made to change the effort of the initial proposal for the integration in Spire with the development of the Kotlin and Swift SDKs.

4. Next Steps

- Example DApp
- Draft implementation in Fearless Wallet
- Draft of individual Packages
  - Beacon SDK – Typescript
  - Beacon SDK – Swift
  - Beacon SDK – Kotlin
- Draft of Message Definitions and Flow (eg. Permission Request)
- Finalization of Message Definitions and Flow
- Support for Beacon KSM in AirGap Wallet

- Release of individual Packages on NPM as beta