Polkadot

Proposal: Cypress Dapp testing plugin and Multix development

Proponent: Tbaut
Beneficiary: ChainSafe Systems 14gaKBxYkbBh2SKGtRDdhuhtyGAs5XLh55bE5x4cDi5CmL75
Date: Aug 2023 - Jan 2024
Requested DOT: 42820 DOT or 214,099 USD using EMA-30 DOT from Subscan on the day of submission
Short description: Multix is a simple interface to manage complex multisigs. It has been live for more than 6 months and its continuous development has been made in the open. It's become the de-facto interface for teams to manage their multisigs, making it easy even with the use of pure proxies. It supports more than 13 chains to date and users' feedback has been overwhelmingly positive. This proposal aims to continue Multix development, as well as to build a Cypress plugin to benefit all Polkadot Dapps.

Currently it's very hard to properly do integration tests with Dapps. Developers have to mock the responses from the wallet and from the nodes. They spend a lot of time developing solutions which provide few guarantees. These strategies and tools are not themselves tried and tested and can lead to a false sense of safety. Multix is potentially vulnerable to this too. As a solution, we propose to build a plugin for the popular testing framework Cypress. This open-source plugin will allow teams to automate Dapp integration tests, without the need for mocking. As we talked to Dapp developers in our network, from Talisman to InvArch, our contacts have shared their enthusiasm.

There is no better way to build a tool than to build it for your own product. This will not only benefit Multix but also the entire ecosystem. This is why this proposal of Multix development is coupled with the building of a Cypress plugin to benefit Polkadot Dapp developers and ultimately Dapps users.

Context of the proposal

ChainSafe Systems was founded in 2017 in Toronto, and is a Canadian-based blockchain development studio with more than 100 employees worldwide. Our vision is to bring the various blockchain ecosystems closer together, building bridges and increasing client diversity by providing alternative protocol implementations. Chainsafe has been building web3 and in particular Polkadot products and infrastructures for 4+ years. We are currently involved in many Polkadot-related projects, from being the core contributor of Sygma, a Substrate to EVM bridge, to building a Go implementation of the Polkadot host called Gossamer. Many team members

are also part of the Kusama and Polkadot fellowships.

Initially funded through the Polkadot treasury for the first 5 months, the Multix team has demonstrated that it was possible to build a multisig interface that is easy to use while keeping it as trust minimized as possible. Multix has been open source from day 1. It does not store the multisig data in a database. The UX is top-notch thanks to a trust-minimized indexer, where any information solely comes from the chain. Establishing a strong presence in the ecosystem, Multix fosters Web3 values and security best practices. Multix paved the way, being the first interface to push for pure proxy use. It was also the first to allow submitting any type of extrinsic with ease. From blog posts to a presentation at Decoded, the team has been pushing the use of multisigs to the next level.

Problem Statement

Multisigs

Many teams in the ecosystem, Chainsafe included, need to manage DOTs securely. While other ecosystems such as Ethereum have very well-known and respected non-custodial multisig solutions such as Gnosis Safe, no such solution existed for Polkadot so far. Businesses and tech-savvy users can create flexible multisigs using proxies, but creating and handling those was very cumbersome, and error-prone.

When using a multisig, you may encounter these issues

- Building a Multisig with Polkadot-js/apps can feel complex and overwhelming
- Need to first add all the signatories and proxies in your address book
- Seeing and reviewing multisig transactions requires external communication between signatories
- The most important part of the process, the extrinsic validation before execution, requires many steps that users may easily avoid doing, putting them at risk of signing unwanted transactions.
- The last user that will submit the transaction needs to copy/paste the extrinsic information (call data).

Multix solves these issues. It has been live since February 2023, and is getting better every day!

Dapp testing

To build resilient Dapps for Polkadot, like Multix, developers need to have automated tests. It helps save time, find bugs early, build faster and cheaper. In contrast to unit testing, in which you test a small isolated unit, integration testing usually involves testing a particular functionality that has dependencies on another functionality (e.g. you submit a wrong extrinsic to a wallet, it gets signed, then you submit it to the node, that rejects it, etc). The goal of these tests is to check the connectivity and communication between different components of the application. It recreates what real users actually do with the application.

Given the high innovation pace of the Polkadot ecosystem, having integration tests for Dapps is a must. Ideally, those tests should reproduce all the common scenarios that the users would face, including errors. In Multix this would for instance include

- creation of a multisig with proxy
- submission of a transaction
- multisig transaction its decoding
- submission by other participants.
-

But also scenarios where

- the user doesn't have enough funds to submit a transaction
- transaction is rejected because it is malformed.
- effective rotation of a multisig signatory, etc.

The vast majority of these are dependent on a wallet and a running node, both of which evolve constantly.

The way developers test these today is either: they actually don't and users are the testers in production, or they create mocks. Mocks are hard-coded values of a given answer to a request, e.g. to a wallet signature. While this works at a certain point in time, such solutions are not future-proof. They generally run against one version of a browser and one version of a blockchain node. As the nodes and wallet evolve, so do the requests and answers. Developers need to constantly change these mocks and the maintenance burden becomes large, while the actual safety they bring is reduced by the fact that they may become obsolete from one day to another.

Solutions & Objective

The treasury proposal is to fund the maintenance and future development of Multix, including a Cypress plugin for integration tests, benefiting all Dapps developers and ultimately Dapps users.

Multix has already pushed the UX boundaries for managing and handling multisigs.

- Creating a new multisig takes seconds.
- Users get access to a multisig with a proxy out of the box allowing them to change the signatories or the threshold at any time without having to move their funds or even communicate about this change.
- Rotate proxy/multisig signatories with ease and high confidence
- New multisig transactions, even the most complex can be made easily from the interface.
- Proposals appear automatically on the interface, the calls are decoded, shown in a user-friendly way, and can be reviewed and signed for easily.
- No need to pass around any data. Signatories don't need to be in contact.
- Multix is open and interoperable. It does not require any api to be connected to it. It is compatible with multisigs created manually, it also supports multisigs without proxies.
- No server other than a Subsquid indexer is used. All the information displayed by the interface comes from the chain.
- View-only mode allows you to follow multisigs you are not part of
- Supports on-chain identity
- Pleasing visual design which is simple and elegant, leading to a great UX
- Mobile friendly. Submit extrinsics or review and sign transactions on the go
- See who signed/needs to sign a transaction
- 13 chains supported at the time of writing, and counting
- Support setup with many proxies and you get an overview of the global setup
- Multix's Github repo and code is state of the art with well-documented and tagged issues and pull-requests, good first issues, testing scenarios.

This list is what Multix offers today. Many feature requests have been made, we list them in the "Implementation & Milestones" section below. The goal is to build as much as has been requested, to cater to most use-cases.

The Cypress plugin that we intend to build will aim to allow developers to
- create arbitrary users in the plugin wallet based on a mnemonic.
- mimic a wallet integration by requesting those users and their addresses.
- answer requests to programmatically sign any type of transaction.

Note that this plugin will actually sign the transactions so that they can be submitted to a node, or the dev instance of a node.

Implementation & Milestones

General project phases, mostly running in parallel

1. Multix Features development and maintenance (see below)
2. Cypress plugin development based on generic Dapps needs
3. Design and UX development
4. Continuous UX interviews and feedback
5. General Multisig education content such as regular articles and presentation
6. Security R&D

Based on the feedback we may change some features. Note that they are not sorted in any particular order and we prioritize the ones that are the most requested.

Multix Features

Features	**Description**
Signing with a multisig from any Dapp using WalletConnect	While Multix is widely used today for one-off extrinsics, e.g. bounty payouts, balance transfers… It would be highly beneficial to be able to use Multix with say the staking dashboard, or to swap assets on a DEX… WalletConnect allows this.
Deep linking to multisigs	Whether someone is part of a multisig or not, users would like to share links to a specific multisig or pure proxy. This way, anyone visiting this link would be able to see the transactions and review/sign if they are part of the multisig.
More parachain support	As parachains win slot auctions and get traction, the need to add more parachain support is never-ending. Our team is dedicated to supporting the parachains that request Multix.
Support parachains without the proxy pallet	Some parachains do not have the proxy pallet, yet they would like to use Multix for their multisigs. It is possible and should be allowed.

Address book	While Multix allows adding custom names to an address. There is no easy way to manage all addresses in one place.
Community integration support	Multix uses an open protocol that can be very easily leveraged by other tools such as the Discord bot built by ChaosDAO. Also, it is retro-compatible with multisig proposals submitted on polkadot.js/apps. The Multix team is dedicated to support community initiatives and guide teams willing to leverage Multix.
Arbitrary extrinsics from Multix directly	Multix already plays well with any multisig proposal made outside of its interface. You can craft an extrinsic in Polkadot-js/apps, paste the extrinsic in Multix and allow you to submit it. However, switching interfaces is not ideal and advanced users may be willing to be totally in control.
Historical transaction	Reviewing who voted when, and on what is possible using an explorer today, but having it nicely displayed in the interface would be greatly appreciated.
Proxy for each signatory	Some businesses have very advanced needs and use cases, requiring 1 proxy in front of each multisig signatory account. Multix can detect and display these accounts for users.

Cypress Plugin Features

Features	**Description**
Create accounts	When launching a test, developers should be able to specify what account the plugin will have. They can pass a set of mnemonics and names.
Connect to a wallet	The plugin will mimic the polkadot-js extension. The Dapp will be able to connect to it, as any wallet sharing the same interface.
Retrieve accounts	This is the first step that any Dapp generally does. It connects to the wallet and retrieves the accounts from the wallet. The plugin will allow this.
Sign transactions	This is the most anticipated feature. Developers should be able to sign automatically any transaction that is presented to the wallet plugin, as long as the user has been initiated.

Documentation	While we intend to build this plugin and use it right away, we want to make sure its documentation is top notch, including examples.

Team

- Developers
- Designer
- Product manager
- QA manager

The initiator and the main developer of Multix is Thibaut Sardan. He is an ex Parity developer who has contributed to major ecosystem tools from being the first developer of Polkassembly, to one of the main maintainers of Polkadot-js extension, Polkadot Vault/Parity Signer, or its UX centric fork Stylo. Thibaut has a great interest in building user-friendly yet complex Web3 products, constantly bringing security closer to usability so that users do not have to choose.

Timeline

This proposal is aimed at funding the team for 6 months, starting from when the proposal passes. While it is hard to predict this far in the future the exact amount of Multix features that will be implemented, we will implement as many features as possible, from the above list. Alongside Multix development we aim to complete the first implementation of the Cypress plugin within the first 3 months.

Payment Details

The requested amount for a 6-month ongoing development grant is 214,099 USD in DOT on the day of submission using the EMA30 rate by Subscan. The allocation is asked for as a single, up-front payment to cover our ongoing development costs and for the team to be able to provide a runway to their developer resources.

The following resource table outlines the expected costs for the Multix team. The table consists of:

- The current base cost to operate the team sums the number of hours for each month at a rate of 90$/h.
- We have several experienced devs working on the projects.

- We expect one dev, PM, Design and QA to work only part-time.
- an 18% operational overhead to cover the costs of running the project as a team within ChainSafe Systems (e.g. operations, offices, equipment, tooling, infrastructure, legal)

Month	Dev h	PM h	Design & UX h	QA h	Base cost	Cost + 18%
1	218.4	33.6	16.8	67.2	$30,240	$35,683
2	218.4	33.6	16.8	67.2	$30,240	$35,683
3	218.4	33.6	16.8	67.2	$30,240	$35,683
4	218.4	33.6	16.8	67.2	$30,240	$35,683
5	218.4	33.6	16.8	67.2	$30,240	$35,683
6	218.4	33.6	16.8	67.2	$30,240	$35,683
Total	**1310.4**	**201.6**	**100.8**	**403.2**	**$181,440**	**$214,099**

Beneficiary Information

The beneficiary of this proposal is ChainSafe Systems Inc, Toronto, Canada. The on-chain address is 14gaKBxYkbBh2SKGtRDdhuhtyGAs5XLh55bE5x4cDi5CmL75. The funds' manager is Hatcher Lipton, COO at ChainSafe Systems, reachable through hatcher@chainsafe.io for any questions regarding the payment conditions.

For any questions about the technical nature of this proposal, you can message me, Thibaut, at thibaut@chainsafe.io