Personal Dao:
A Non-custodial, private, peer-to-peer treasury & data storage solution

Written and Founded by Jesse D. Williams Jr.

September 10th, 2023

Motivation

Bitcoin's introduction to the world following the 2008 global financial crisis proved that a peer-to-peer, borderless, transparent digital currency is possible. It birthed the cryptocurrency industry and since then, the growth of the industry has been tremendous. Cryptocurrencies have proven how useful they are for the average retail investor looking to safeguard their wealth in a manner that preserves their sovereignty. However, cryptocurrencies are just as complex as they are useful. The average person lacks the technical knowledge necessary in order to take full advantage of the sovereignty that cryptocurrencies offer. It didn't take long for centralized institutions to notice and exploit this apparent knowledge gap. Exchanges, banks, wallet providers and protocols have begun to offer crypto services in which the user relinquishes at least partial sovereignty of their funds over to these institutions. This completely defeats the purpose of cryptocurrencies and leaves retail investors subject to the same risks that drove them to flee the traditional finance industry in the first place.

Exchanges are infamous within the crypto industry for offering services in which custodianship rests with them- the intermediary. Many crypto wallets and protocols market themselves as non-custodial and therefore a sovereign alternative to these exchanges. They boast about the fact that they don't store the cryptographic keys needed in order to authorize on-chain transactions. They use this fact to justify the claim that their services are non-custodial. While it may be true that these protocols don't store your cryptographic keys, it doesn't certify them as non-custodial. MetaMask, for example, doesn't store cryptographic keys; however, they do store the names, dates of birth, contact information, wallet addresses, home addresses, telephone numbers, interests, credit card information, bank accounting/routing numbers and financial assets data for each of their users (here is a link to the privacy policy MetaMask currently implements: https://consensys.io/privacy-policy/). Even Though MetaMask holds custody of an invasive amount of data for their users, they still claim to be a non-custodial service merely because they refrain from storing a single piece of data: the users' private keys.

Perhaps the most important object of custodianship that the overwhelming majority of crypto projects never mention is the actual code that facilitates blockchain transactions. Consider the following:

1. The UI (User Interface) of an application is the component that allows users to be able to interact with the APIs (Application-Program interfaces) that manage their data in a manner that requires the user to have no coding experience. This component is critical for the vast majority of app users since most users have no idea how to interact with servers directly. They rely on User Interfaces.
2. The API of an application is a component that allows user-generated data to be securely manipulated prior to being stored on the servers where the data will stay until erased or changed by the users. This component is critical for anyone (including people with coding experience) who doesn't have physical access to the servers where the data is stored.

The UI and API are absolutely necessary in order for a user of a crypto wallet to be able to access the funds within their wallet. Wallets like MetaMask, who claim to offer non-custodial services, maintain full custodianship of both the UI and the API components. As a consequence of that, the User Interface & Application Program interface of MetaMask both serve as a major point of centralization while simultaneously cementing metamask as an intermediary of every single transaction that is administered by MetaMask. When bad actors execute mass hacks on wallets that are supposed to be non-custodial, they do so by targeting the UI and/or API of the intermediary that has custody of them. I cited MetaMask as a reference, but the information I've provided here applies to every crypto wallet in the industry with the exception of **Personal DAO**.

True crypto proponents often look to DeFi protocols as a place to achieve maximum sovereignty of funds. There are two big problems with relying on DeFi protocols for storing value:

1. DeFi protocols rely on an architecture in which substantial sums of value are stored within a single protocol- making them a gold mine for hacker groups & malicious states looking to execute mass hacks. Any user storing their funds on traditional DeFi protocols do so at heavy risk to their assets.
2. DeFi protocols often feature UIs and APIs that are in the custody of the organization/foundation that spearheads development. As a result, the same vulnerabilities that MetaMask has exist within DeFi protocols- with the added risk regarding the fact that the DeFi protocols are open-sourced.

In the end, DeFi protocols are often counterproductive for users who wish to safeguard their crypto while maintaining sovereignty and privacy of their funds.

Solution

In response to the opaque nature of the traditional finance industry, as well as the misleading & risky nature of “non-custodial” wallets & DeFi protocols, I’ve innovated a software architecture that enables users to have true custodianship of the entirety of the application that they rely on for sending, receiving & storing their cryptocurrencies & private data. When users purchase their own Personal DAO, they receive custodianship of the User Interface, The Application-Program Interface and all smart contracts critical to the functionality of the application. Each Personal DAO is completely disjoint from all other Personal DAOs that exist. This means that two people who purchase their own respective Personal DAOs will have no point of centralization between their assets/data.

Unlike MetaMask, where every user shares a single UI and API that they use to access their individual wallets, with personal DAO, every instance has its own UI and its own API that is used to access only the assets of the users who have been permitted to enter that particular Personal DAO. Consequently, the funds stored in Personal DAOs are never accumulated into a single protocol. This means that every Personal DAO deployed to the Internet is its own individual protocol. This sort of architecture is inherently safer than all other wallet/DeFi protocols currently in existence because it makes it impossible for mass hacks to be carried out. Even in the event that a bad actor finds a way to breach Personal DAO security protocols, that bad actor *still* would be unable to execute mass hacks against Personal DAO users because each DAO is a separate entity that must be looted one by one. This makes targeting Personal DAOs much less lucrative- resulting in a project that is intrinsically safer than all other competitors.

Use Case

The crypto industry has created many successful, peer-to-peer equivalents to the services that are most important within the tradFi industry(i.e. trusts, treasuries & lending). Personal Dao aims to take peer-to-peer financial services a step further by providing retail investors with a tool that they can use to pool, manage, and leverage funds within their own, interpersonal communities while preserving the sovereignty of their community’s funds 100%. Anyone who wants to be the founding member to their very own Personal Dao is able to do so. As the founding member, you’re able to invite whomever you wish to become a fellow member of *your* Personal Dao. Unlike other Daos that are available to anyone and everyone in the world, your Personal Dao is only available to those who are permitted to enter by you, the founding member.

This means that the Dao participants only consist of people that share **your** interests, values and community. When you use your Personal DAO for peer-to-peer financial services, you borrow funds against your *collateralized* assets; it's **your** community that you borrow from- consequently, it's **your** community that you enrich upon servicing your debts.

Personal Dao does more than just offer financial services- it's also a data storage solution. Every member of your Personal Dao receives their own canister smart contract in which they are able to keep a diary, secure notes, a private crypto wallet, an address book, a file drive and more. This means that in addition to storing things of monetary value, your community will also be able to store things of sentimental value, all in one space. Your Personal Dao will be able to transcend generations- allowing those who follow you to inherit their history as well as the wealth that their predecessors accrued.

In addition to offering peer-to-peer financial services and secure data storage, personal DAO offers a means by which communities within the Internet Computer's ecosystem can organize, deliberate and vote- all on one accord, in the best interest of the collective. The purpose of this white paper is to explain to you exactly how each Personal Dao works.

The Technical Details:

Personal DAO is hosted completely on the Internet Computer Protocol Blockchain. The application consists of five different smart contracts responsible for housing each of the five components of the application. Those components are as follows:

1. **User Interface Canister**: This is the component of the application that users interact with directly. The web address, buttons, pictures, textboxes and anything else that a user directly encounters while interacting with the application is all stored in this section of the app. In lay person's terms, it is the face of the application.
2. **Backend Canister**: This component of the application is responsible for keeping record of which user principals have access to the app, which canisters belong to which user, which principal is the admin of the application and many more things. It's worth mentioning that the backend canister is responsible for delivering upgrades to the manager canister as the manager canister cannot upgrade itself. In lay person's terms, it is the brain of the application.
3. **Manager Canister**: This component of the application is responsible for communicating with a canister that is responsible for storing new Personal DAO updates(aka the upgrade store canister). Whenever new updates are made available, the DAO members get to decide whether or not they wish to implement the new updates. In the event that consensus is reached in favor of implementing

the updates, the manager canister is responsible for retrieving those updates from the upgrade store canister and then promptly delivering those updates to the rest of the canisters within the protocol.

4. **Treasury Canister**: This component of the application is responsible for storing the assets deposited by the DAO members as contributions to the DAO. This component keeps track of each dao member's respective contributions, and computes the voting power that they are due as a result of their contributions.
5. **User Data Canister**: This component of the application is responsible for storing DAO members' individual data and assets. Every member permitted to enter the DAO will have a unique User Data Canister created solely for their individual data usage, that is deployed to the Internet Computer.

For each Personal DAO that is deployed to the internet computer, there is a single corresponding Founder's NFT. Each Personal DAO is configured to recognize its corresponding Founder's NFT as well as the principal of the wallet that holds said NFT. The holder of the Founder's NFT is given administrative privileges. Tranferral of the Founder's NFT results in transferral of administrative privileges. A separate article detailing the administrative privileges will be released at a later time.

There are many instances of Personal DAOs deployed to the Internet Computer. Each Personal DAO that is deployed has its own URL, UI canister, Backend Canister, Manager Canister, Treasury Canister, and for each member within that Personal DAO, there is a User Data Canister.

Each backend canister is assigned as the controller for:

1. The backend canister (Itself)
2. The UI canister
3. The manager canister
4. The treasury canister
5. All user data canisters

Each manager canister is assigned as the controller for:

1. The backend canister
2. The UI canister
3. The manager canister(Itself)
4. The treasury canister
5. All user data canisters

Note: both the backend canister and the manager canister are controllers of all the canisters that comprise the Personal DAO, including themselves respectively.

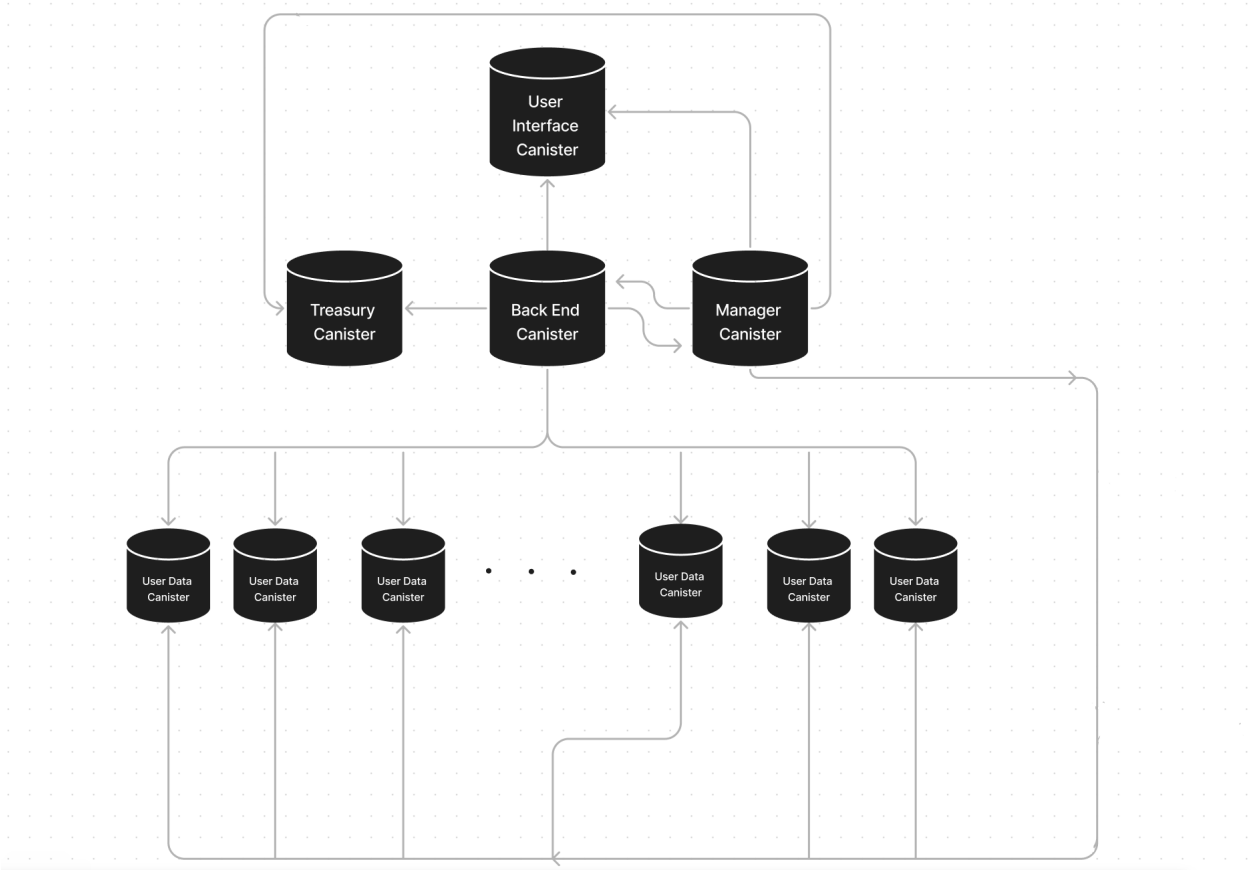
The backend canister is coded with functions that allow it to perform all canister controller operations upon itself, as well as upon all other canisters that it controls within the application. The manager canister is coded with functions that allow it to perform upgrades onto the backend canister whenever upgrades are available.

Note: The canisters will only execute ***canister controller operations*** *if they come from the principal that owns the corresponding Founder's NFT or if consensus among dao members is reached first. This leaves access to PersonalDAOs' controller settings in the sole custody of the respective DAO members.*

To access owner privileges, the owner of the Founder's NFT signs into the application using the wallet that holds the NFT. The backend canister verifies that the principal signing into the app is the same principal that owns the NFT.

Canister Architecture:

The diagram below illustrates the canister controller settings for an instance of Personal DAO deployed to the Internet Computer. The arrow indicates that a canister controls another, with the canister on the pointed end of the arrow being the controlled by the canister at the opposite end of the arrow.



Update Propagations:

The diagram below illustrates the mechanism by which Personal DAOs are updated while preserving user sovereignty and canister controller settings of each individual Personal DAO. The Updates are directly uploaded to the master copy, which the dev team maintains control over. From there, the Upgrades Store canister pulls those updates from the master copy and stores them. The other Personal DAO copies deployed to the internet computer may then make a request for those updates and then install those updates to themselves.