This is a WIP

Wallet Kernel & dApp Architecture

7 May 2025

Digital Asset

This is a WIP

Overview

Problem Statement

dApp / Wallet Kernel / Trust Zones

Functionality, APIs and Authentication

Wallet Kernel Deployment Options

This is a WIP

Problem Statement

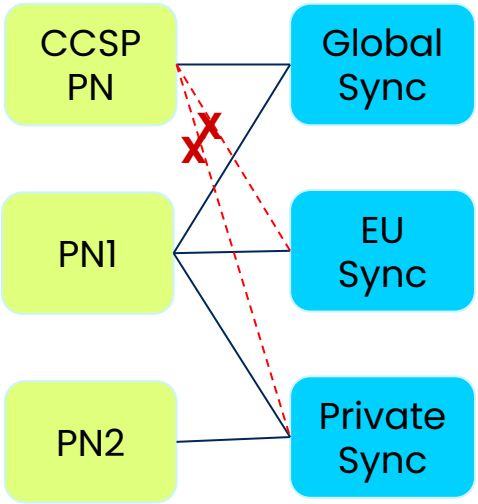
What problem are we trying to solve?

This is a WIP

Privacy vs Wallet Architecture in Web3

Standard Wallet Architecture Does Not Work in Canton

- Existing Web3 Wallet infrastructure and dApps largely assume **total state replication**.
- As such, CCSPs are used to submit transactions directly to the network.
- dApps read the **global state** from an indexing service.
- Canton provides sub-transaction privacy and multiple synchronizers. This leads to a **globally segregated state** without the ability to connect to every sub-network for reading and writing.
- We need to amend the architecture to support both **generic dApps** in the Canton Network **with key custody**.



Digital Asset

This is a WIP

Canton dApp / Wallet Assumptions

Standard Wallet Architecture Does Not Work in Canton

- The key custody architecture should work **across public and private synchronizers** equivalently without the CCSP to have their own node in every network. (A CCSP may choose to not support this but then their solution would be tied to a specific network).
- Canton dApps must be able to leverage **existing** key custody **infrastructure without dApp specific** changes.
- dApps need to deal with the fact that **some ledger data may not be known** to the app provider.
- Users using key custody **do not want to deploy custom backends** for each dApp they use, but they do not want to compromise on security and trust.

We need to solve key custody in a **network of networks with privacy.** This is a different problem than other networks have.

This is a WIP

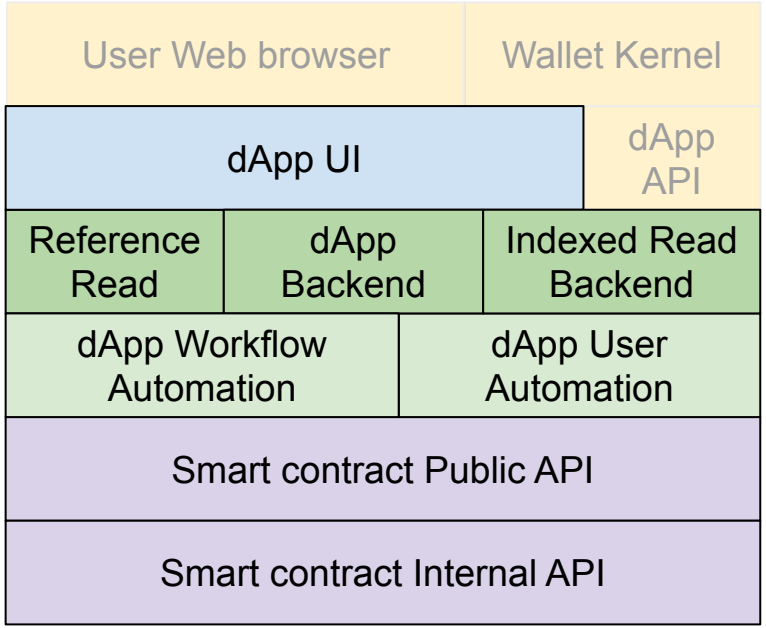
dApp / Wallet Kernel / Trust Zones

Looking at Canton Apps From the Perspective of Trust Zones and APIs

This is a WIP

What is a dApp?

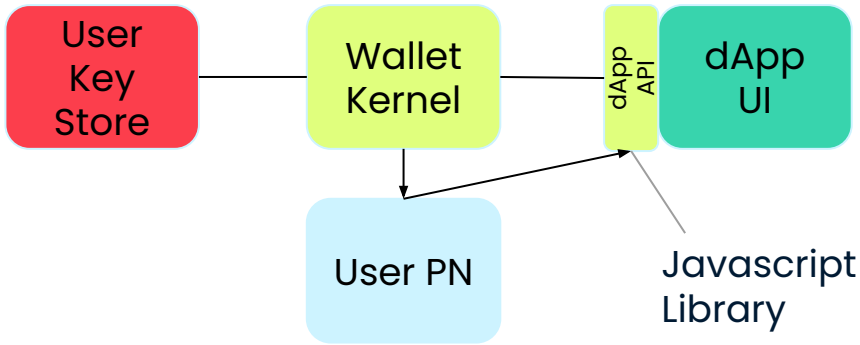
High-level components of a dApp



This is a WIP

Wallet Kernel

Intermediates between dApp UI, User Key Store (CCSP), and Participant Node

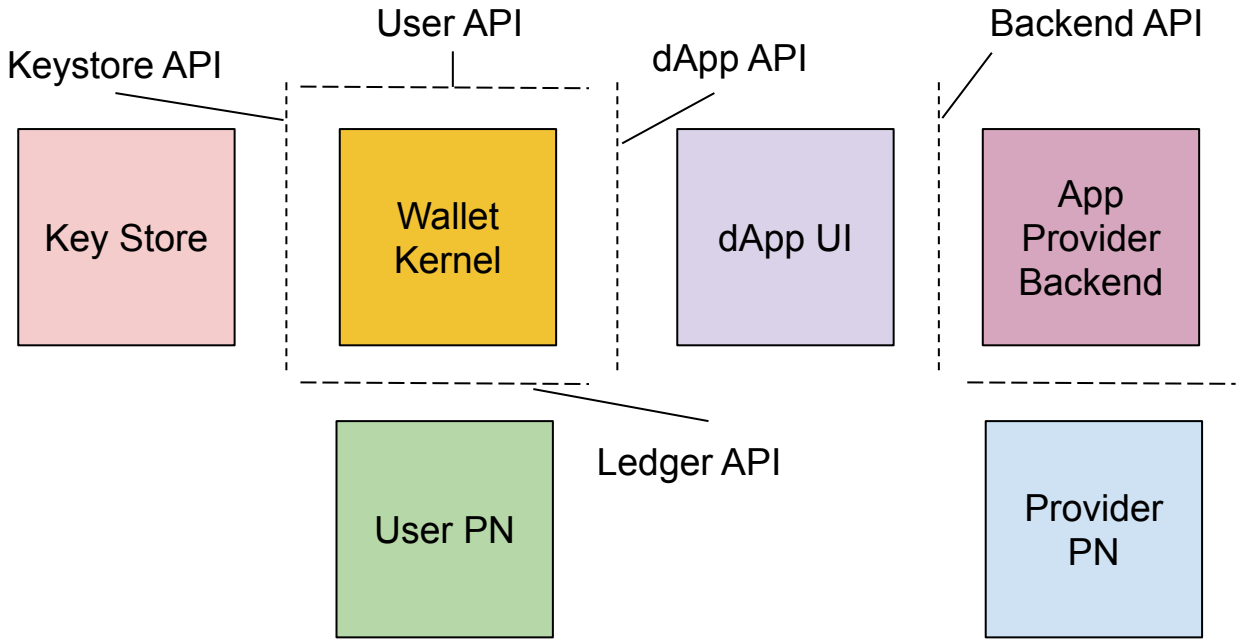


- The **user trusts the wallet kernel**, trusts the key store to not perform unauthorized signing and trusts the participant in terms of reading state. The user doesn't trust the dApp UI and therefore wants to selectively control read and write operations.

This is a WIP

Wallet Kernel

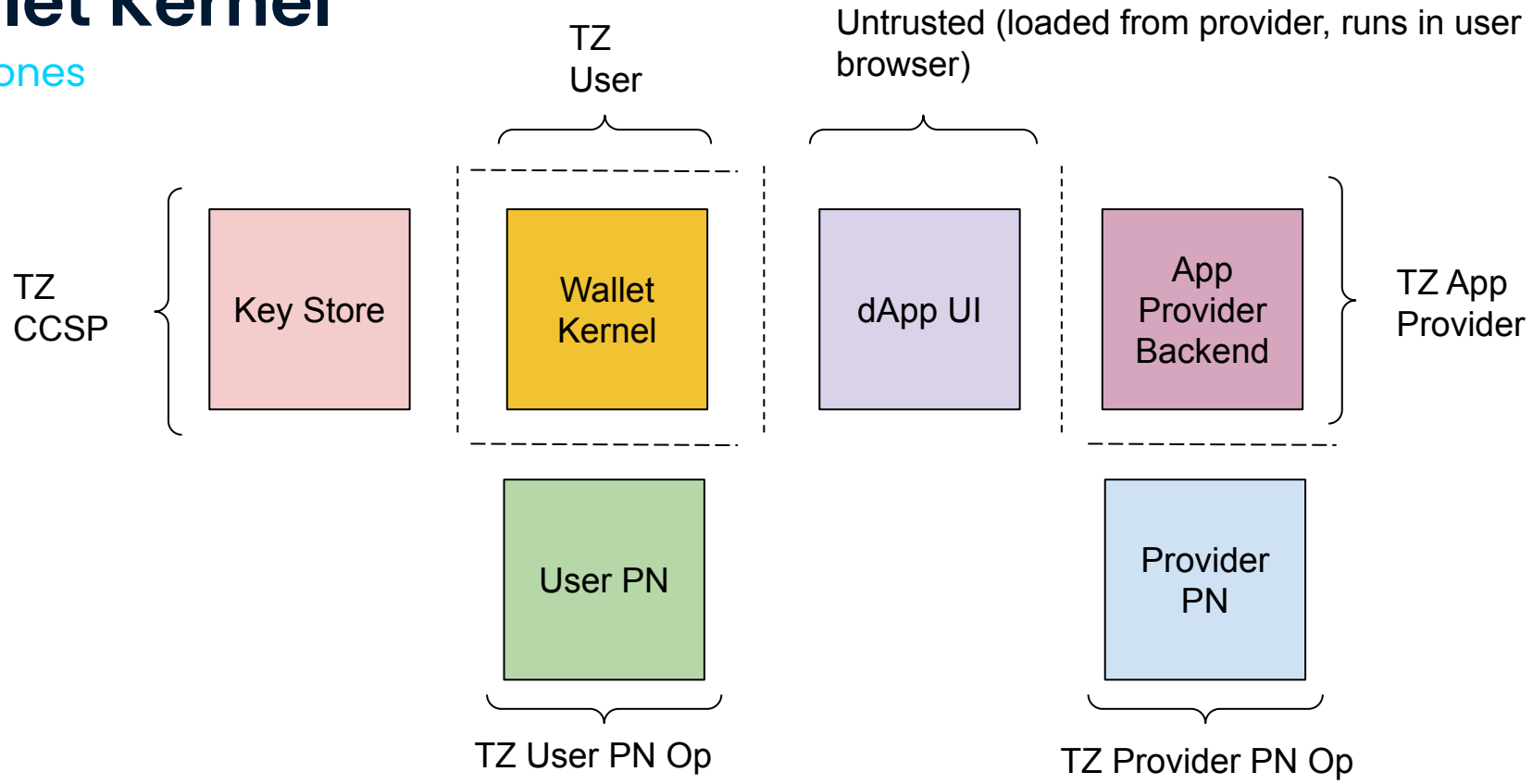
APIs



This is a WIP

Wallet Kernel

Trust zones



This is a WIP

Functionality and APIs

Functionality of the Wallet Kernel

This is a WIP

dApp API

Functional Scope of the dApp API

- **Authenticate:** Trigger the authentication of the dApp UI towards the Wallet Kernel
- **Topology read:** Return parties, participants, and installed DAR packages
- **Ledger read:** Read Ledger State for a given party
  - GetActiveContracts
  - GetContractById
  - GetUpdates
  - GetUpdatesById
- **Contract write:** Submit a command to the Wallet Kernel for it to be prepared and signed
  - SubmitAndWait
  - PrepareSignAndReturn

This is a WIP

Key Store API

Functional Scope of the KeyStore API

- **Authenticate:** Trigger the authentication of the Wallet Kernel towards the Keystore such that the Wallet Kernel can interact with the Keystore.
- **Sign:** Submit a blob, a hash, the target key together with a reference for signing. Expect a signature back.
- **List Keys:** List existing keys
- **Capability:** Available schemes; whether key creation via API is possible; whether key store provides write review.
- **Create Keys:** If supported, trigger the creation of a key

This is a WIP

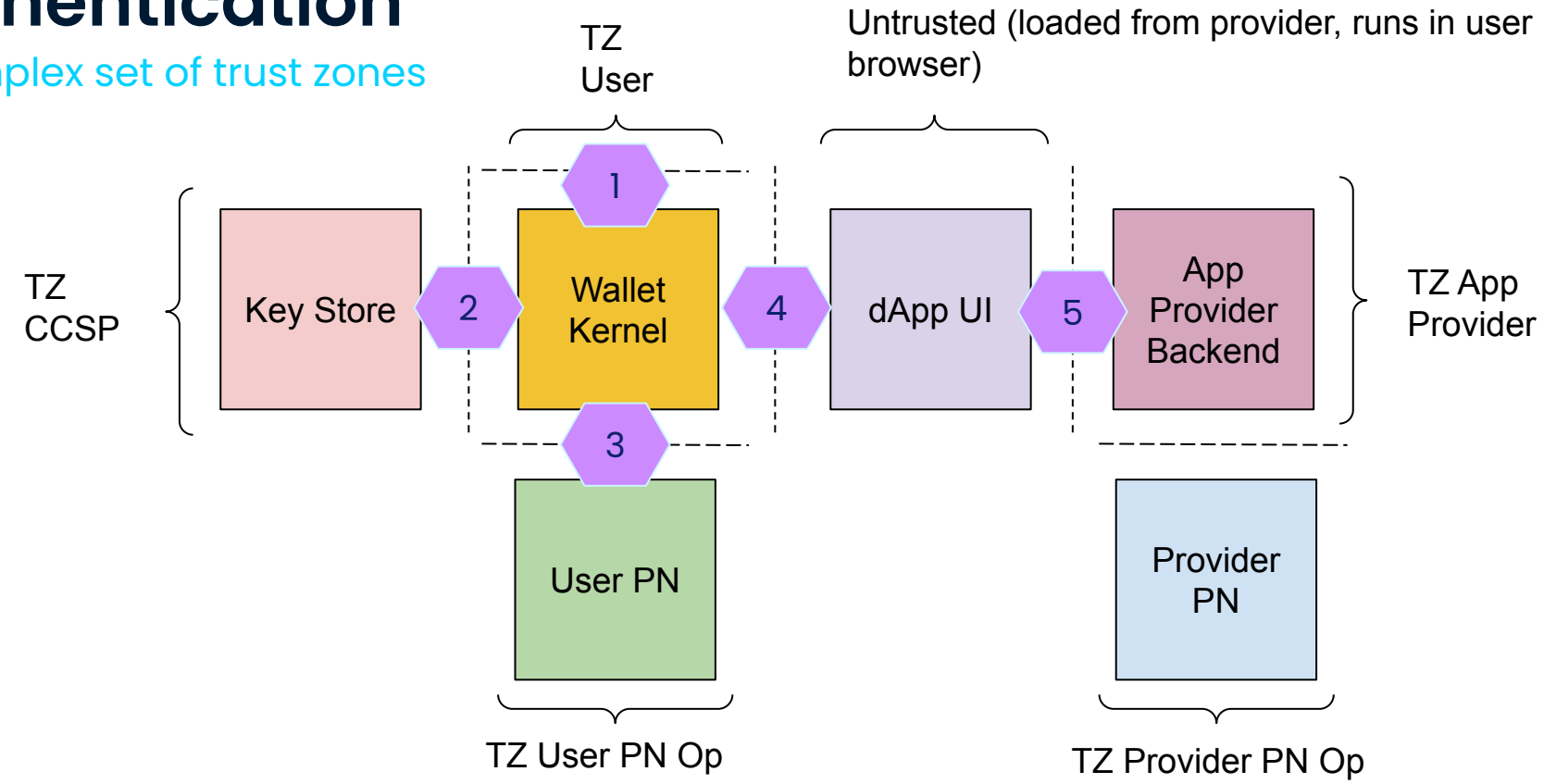
Authentication

High Fences make Good Neighbours

This is a WIP

Authentication

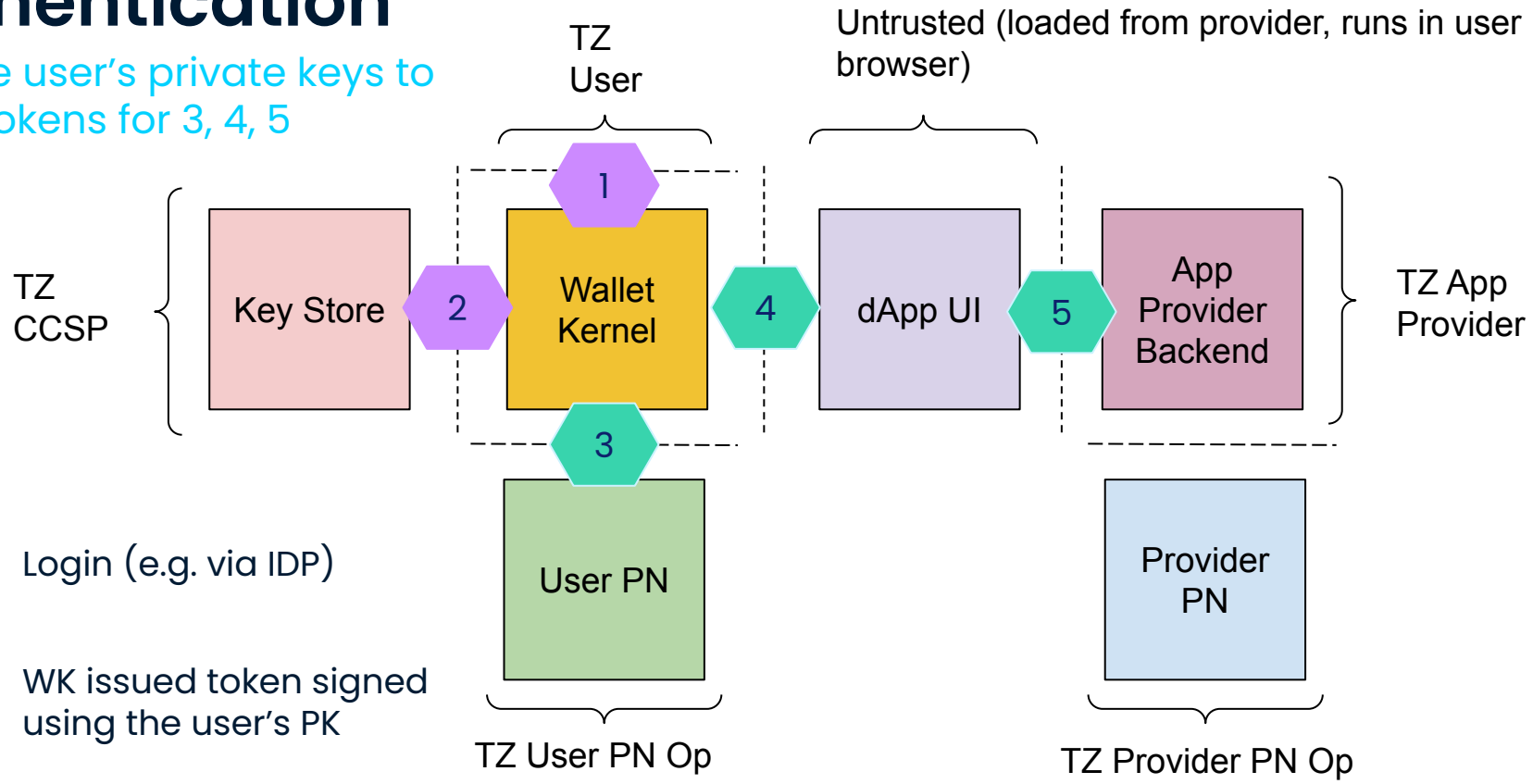
A complex set of trust zones



This is a WIP

Authentication

Use the user's private keys to issue tokens for 3, 4, 5



This is a WIP

Self-Issued OpenID Connect

What is Self-Issued OpenID Connect

- The Wallet Kernel will take the role of the IDP with [self-issued id tokens]
- The user effectively proves “ownership of a key”.
- All participants can verify that a certain key supports api authentication for a given set of parties (using `PartyToKeyMapping`)


```
{
  "iss": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCcd-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs",
  "sub": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCcd-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs",
  "aud": "https://client.example.org/cb",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "sub_jwk": {
    "kty": "RSA",
    "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAt
    VT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMstn64tZ_2W
    -5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2QvzqY368QQ
    MicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbISD08qNLyrdk
    t-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqbw0Ls1jF44-cs
    FCur-kEgU8awapJzKnqDKgw",
    "e": "AQAB"
  }
}
```

This is a WIP

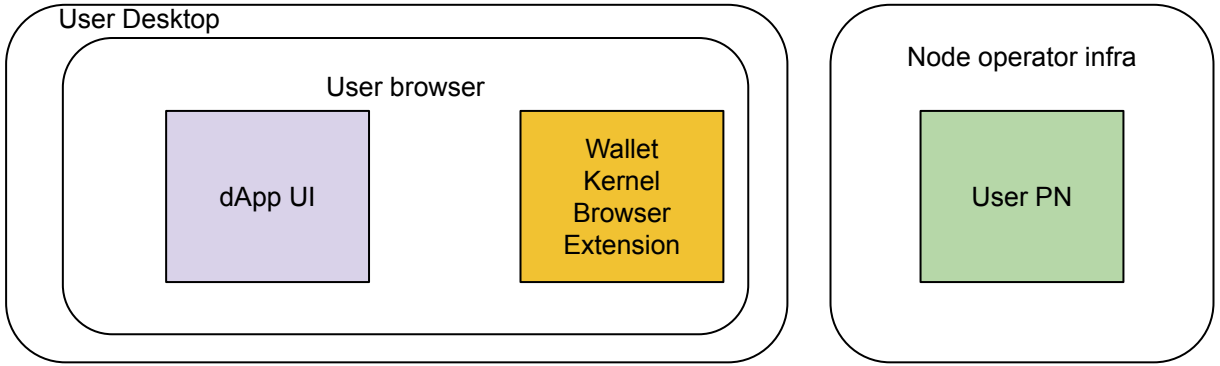
Deployment Options

There are different options

This is a WIP

Browser Plugin

One Plugin To Rule Them All

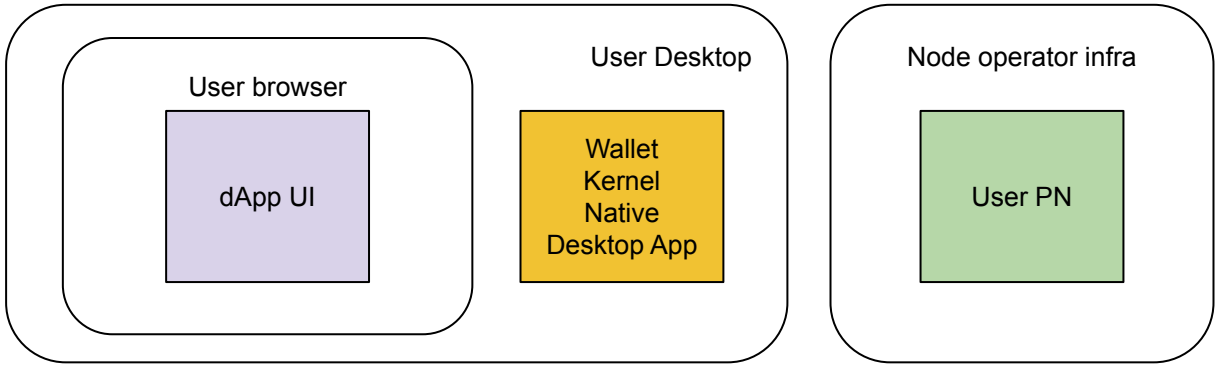


- Wallet Kernel runs as Browser Extension, distributed via Chrome and/or Firefox Addons with Cross-browser support provided by Manifest V3
- dApp communicates with Wallet Kernel via an injected provider (ERC-1193 compatible), made available under *window.canton*
- Uses VanillaJS to provide browser-native UX
- Relies on extension storage to maintain its state

This is a WIP

Desktop App with Protocol Handler

Great UX but Painful to Implement Across all Platforms

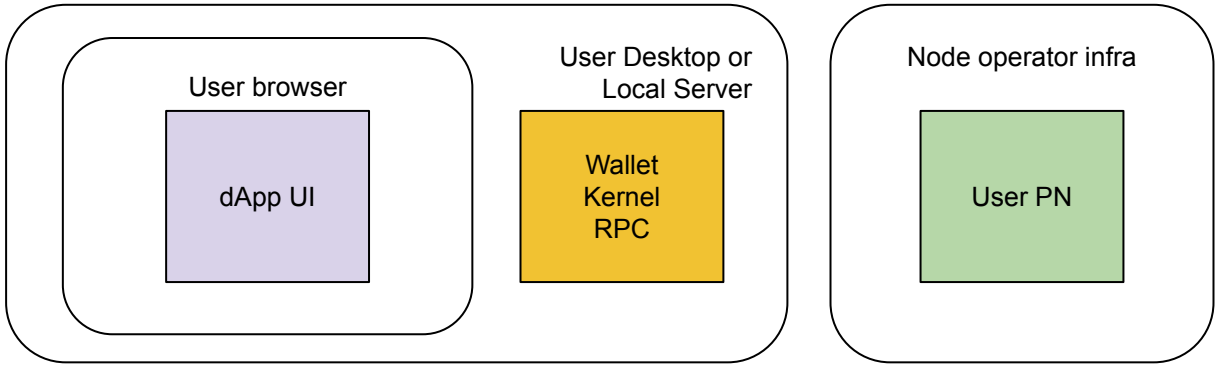


- Register protocol handler canton:// through natively installed desktop app (Windows registry, OSX Info.plist, Linux xdg-mime, Electron App
- Use custom desktop app features to provide native UX (means we need to do it for all OS)

This is a WIP

RPC Service

Operate the Wallet Kernel as a local or remote service



- Wallet Kernel runs as stand alone RPC server somewhere accessible from the local network of the user.
- This can be a remote web-service as long as the kernel can connect to the participant node.
- This can also be a background process on the users desktop, letting the user interact through the browser (towards wallet.localhost).

This is a WIP

CCSP Native Wallet Kernel

Combining Wallet Kernel and Key Store API to Optimise UX

- We need to optimise for both:
  - **Initial delivery** where CCSP is a glorified key-store vs.
  - Subsequent full **CCSP support**.

- The **Core API** allows to **configure a Wallet Kernel** consisting of multiple core modules. Our reference application is one possible configuration.
- A **CCSP** could define its **own configuration** with alternative module implementations or reuse (parts of) our reference.
- For the **JS dApp SDK** to be compatible, the **reference dApp API** module must be used.