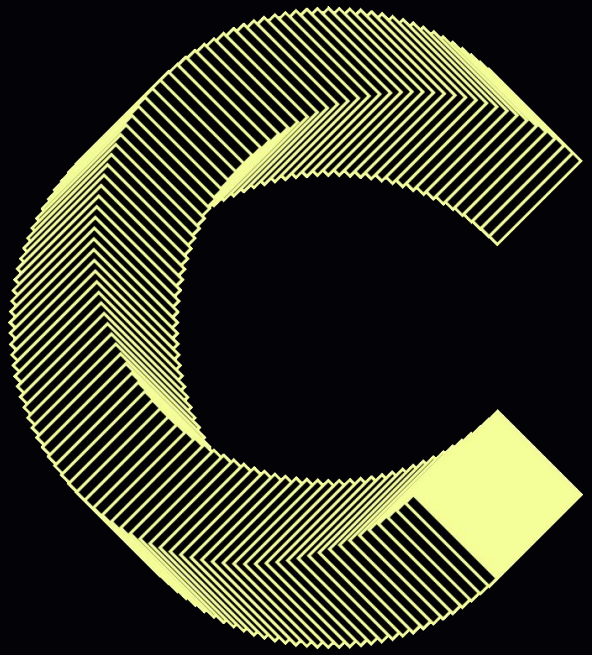
Connecting to Canton Network

Infrastructure considerations and setup recommendations



A guide for market participants

Contents

This guide aims to enable business executives to make targeted infrastructure decisions to connect and get started on Canton Network.

RECOMMENDED READING

Section 1

Canton Network: Key components & deployment considerations

1a Components

1b Deployment

1c Solutions

NOTE: IF YOU'RE LOOKING FOR A QUICKSTART TO CONNECT TO CANTON NETWORK JUMP TO TURNKEY WALLET PROVIDERS

OPTIONAL DEEP DIVES

Section 2

Setup considerations: DAR, Node & DApp dependencies across wallet options

Section 3

Canton Network information flow by chosen setup option

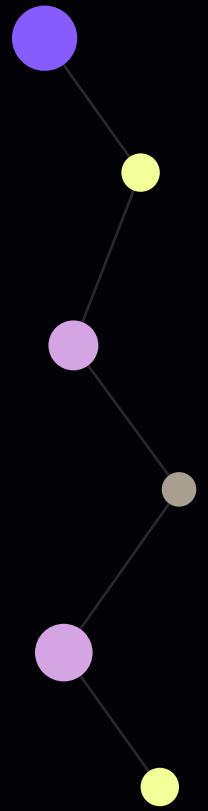
Section 4

Canton Network DApp best practices

Canton

Section 1

Canton Network: Key components and deployment considerations



SECTION 1A

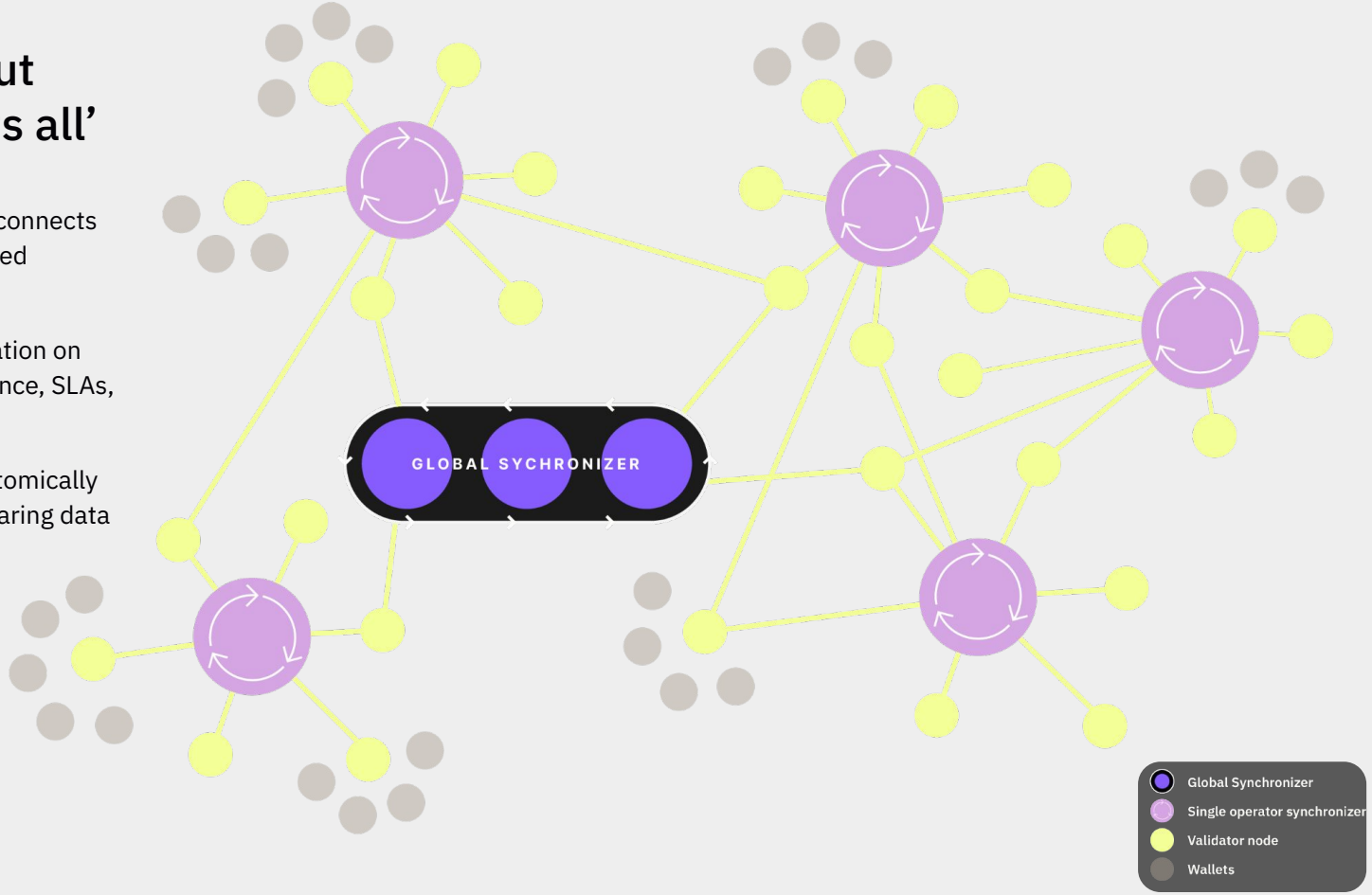
Canton Network’s key components

One network but not ‘one size fits all’

Canton is the protocol that connects many independently operated applications.

Each smart contract application on Canton has its own governance, SLAs, and operating policies.

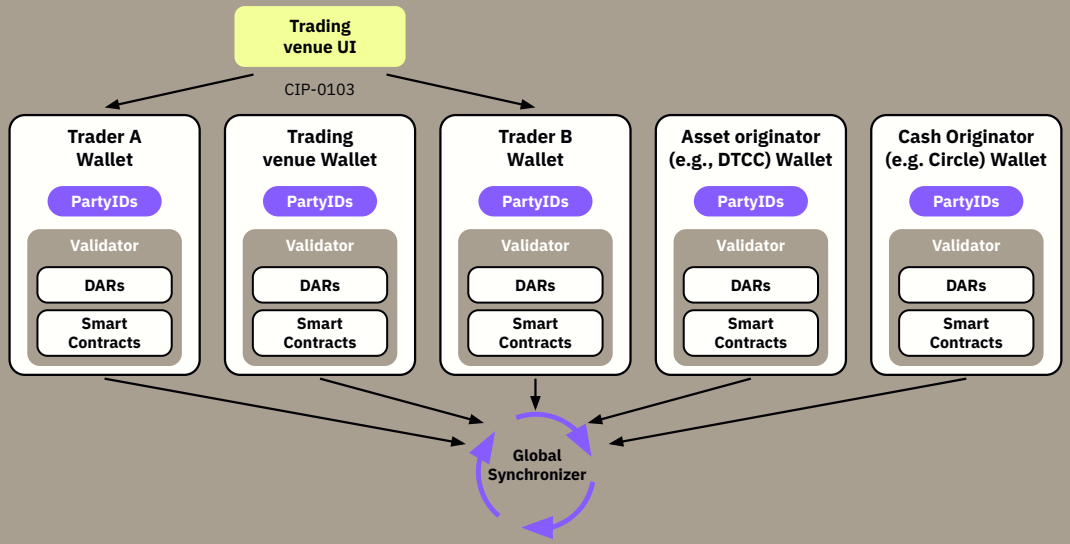
Applications interoperate atomically across the network, only sharing data between participants on a need-to-know basis.



Key terms

PartyID	Unique digital identity assigned to a specific individual, organization, or role
Daml	Canton smart contract language that is used to build Canton applications
DAR (Daml Archive)	The packaged code file containing business logic. It dictates: – **Action rules:** which PartyID can participate in each part of the workflow – **Privacy rules:** what transaction data each PartyID is allowed to see
Smart Contract	The active digital agreement, an instance of a DAR-provided template, that automatically governs the relationship and transactions between involved PartyIDs.
DApp	A Distributed Application (DApp) is a business app on Canton enabling you to securely transact and share data with other participants.

Core components and connectivity



Identity and custody: Every participant holds unique **PartyIDs** managed by their **Wallet**. The Wallet is powered by a **Validator Node**, which stores your private transaction data.

The logic (Daml and DARs): Validators run **Smart Contracts** (derived from **DARs**) written in the **Daml** language to enforce the rules of assets and trades.

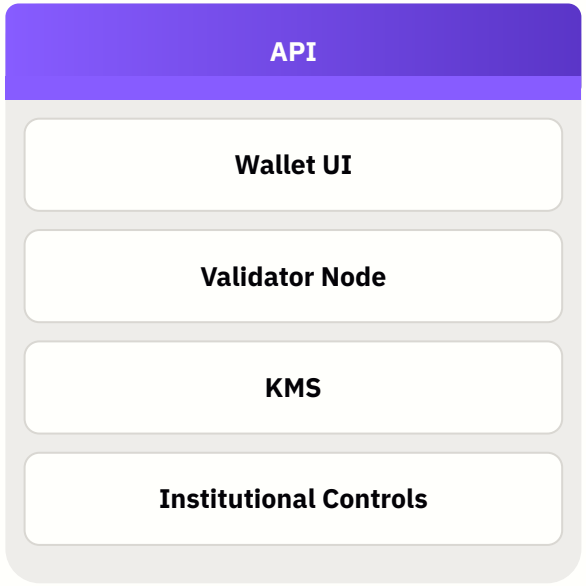
The assets: Asset and cash originators issue digital tokens (e.g., US Treasuries, USDCx) onto the network, adhering to Token Standards (CIP-0056 and CIP-0112).

The network (Canton): All Validator Nodes communicate securely via the **Global Synchronizer**, which guarantees consistency and atomic, near real-time settlement across wallets.

The trade: To execute a trade, Traders A & B connect their wallets to a Trading Venue's DApp UI or API via a standard connectivity protocol (i.e. CIP-103).

Wallet components

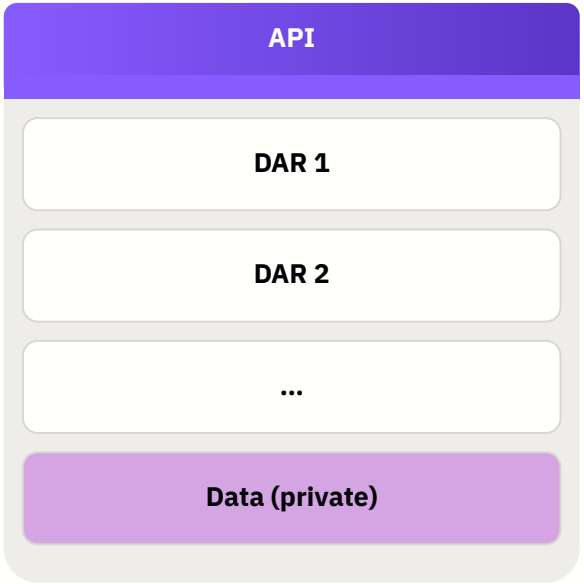
A Wallet on Canton means the secure software your organization uses to manage digital assets, identities, and cryptographic keys on the network.



- **Wallet API:** the integration bridge connecting your Wallet to external business applications (DApps) and other systems
- **Wallet UI:** the dashboard where authorized personnel view assets and initiate, approve, or reject transactions
- **Validator Node:** your secure operating presence which submits and validates transactions for your PartyIDs, and maintains your private view of the ledger
- **Key Management Service (KMS):** the secure vault for your private cryptographic keys, proving ownership and authorizing trades
- **Institutional Controls:** optional governance capabilities, including approval workflows (e.g., dual-approvals for high value transactions), automated risk controls, AML, and compliance reporting.

Validator Node

A Canton Validator Node is the software an organization runs (or uses as a managed service) to connect to the network, execute transactions, and maintain a private view of the ledger.

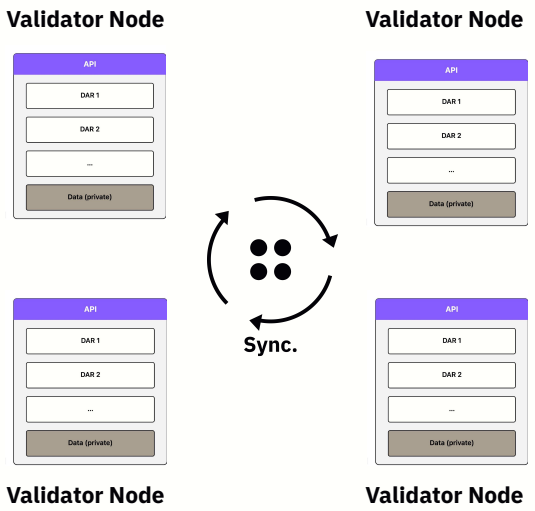


Access to a Validator Node on Canton provides:

- **A secure operating presence on the network:** hosts your digital identities (PartyIDs) and application logic (DAR files)
- **A private ledger view:** stores only the contracts, assets, and transaction history your organization is explicitly entitled to see
- **Transaction submission:** used to submit transactions on your behalf to the network (i.e. to Synchronizers) while ensuring strict compliance with application business rules
- **Transaction validation:** receives transactions ready for confirmation from the network, ensuring strict compliance with application business rules, in particular to avoid double spend
- **Network synchronization:** coordinates securely with other nodes to enable atomic transactions, preserving privacy and eliminating reconciliation.

Global Synchronizer

The shared, decentralized backbone that securely coordinates transactions and contributes towards privacy guarantees and consistency. Think of it like a postal system: it accurately routes the envelopes without ever reading the letters inside.

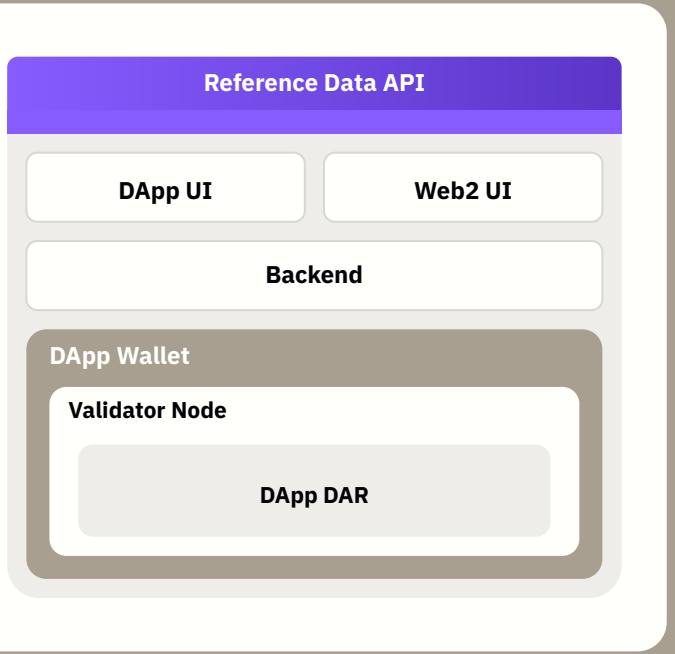


The Global Synchronizer delivers:

- **Connectivity:** provides the shared infrastructure linking Validator Nodes and facilitating cross-app smart contract composability
- **Absolute privacy:** routes and coordinates messages without ever accessing or exposing your underlying business data
- **Atomic execution:** ensures multi-party transactions settle simultaneously (all-or-nothing), eliminating settlement risk
- **State synchronization (Zero reconciliation):** guarantees all participants share a consistent ledger as golden source of truth, eliminating the need for post-trade reconciliation.

DApp providers

An entity that builds, deploys, and maintains a DApp for network participants. At a minimum, a provider supplies the core smart contract logic (DAR) and the user dashboard (DApp UI).



Core DApp components

- **DApp UI (User Interface):** the application interface through which users view data and initiate business transactions
- **DAR (Application Code):** the packaged file containing the DApp's business logic, smart contract workflows, and privacy rules.

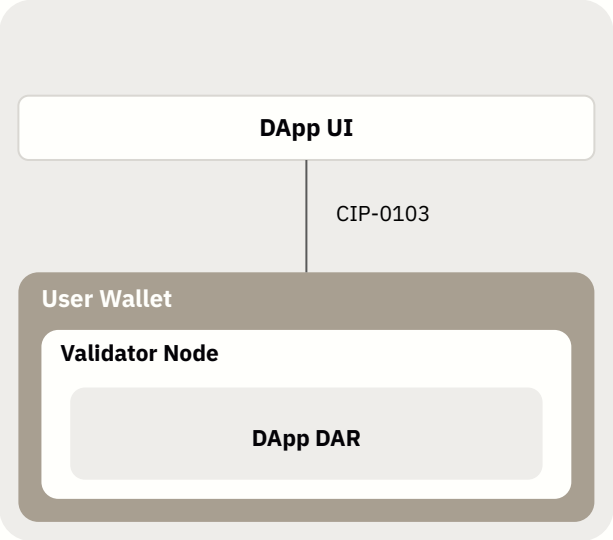
To support sophisticated, enterprise-grade use cases, complex DApps may also include:

- **Reference Data APIs:** secure integration feeds to pull external, real world information (e.g., pricing or identity data) into the application
- **Traditional Web Interfaces (Web 2.0 UI):** standard web 2.0 dashboards for users to manage off-chain business processes, reporting, admin
- **Off-Chain Backend:** traditional infrastructure used to process complex proprietary business logic and store data that does not require distributed network consensus
- **DApp Wallet:** a dedicated, programmatic backend wallet used by the application itself to automatically authorize and trigger system-level transactions.

Note: the DApp Provider is responsible for securely hosting and operating all UIs, APIs, and off-chain backend infrastructure.

DApp and DApp connectivity standards

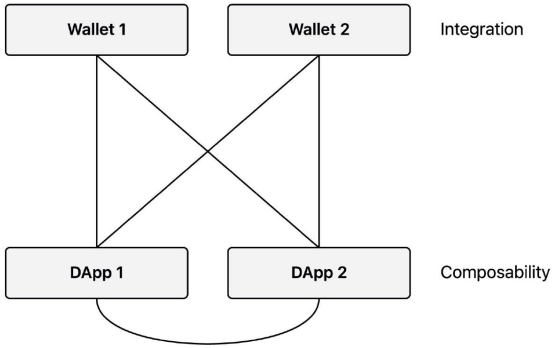
A Distributed Application (DApp) is a business app on Canton enabling you to securely transact and share data with other participants.



Connectivity standard

- **DApp connectivity (CIP-0103 Standard):** ensures plug-and-play interoperability, allowing enterprise wallets and DApp UIs/APIs to connect securely out-of-the-box
- **The DApp UI (Originator):** implements the client specification to universally route requests to any compliant wallet
- **The user wallet (Approver):** implements the server specification to safely authenticate and process those requests.

Canton Token Standards



Token Standards matter for:

- **Accelerated institutional onboarding:** guarantees out-of-the-box compatibility with institutional wallets and qualified custodians (e.g., BitGo, Fireblocks, Blockdaemon)
- **Ecosystem composability:** ensures third-party applications, trading venues, and lending protocols can seamlessly read, hold, and interact with tokenized assets on day one.

The Canton Standards Framework (see details here):

- CIP-0056 **(Token Standard V1):** essential smart contract primitives for basic asset holding, transfers, and atomic DvP settlement
- CIP-0112 **(Token Standard V2):** the upgrade to handle complex market structures, advanced privacy, and traditional accounting.

TOKEN STANDARD CIP-0056

CIP-0056 token standard V1: Standard functionality

Essential token operations

- **FoP (Free-of-Payment) transfers:** enables direct, peer-to-peer token transfers between institutional wallets
- **Privacy-preserving ledgers:** native support for balance querying, transaction history, and two-step transfers with sender/receiver controls

Atomic Delivery-vs-Payment (DvP) token allocations

- **In-wallet asset locking:** institutions securely “lock” assets to a specific trade while retaining 100% custody in their own wallet
- **Escrowless atomic settlement:** third-party applications can execute simultaneous (atomic) DvP swaps — eliminating settlement risk without a trusted central escrow or third-party custodian.

TOKEN STANDARD CIP-0112

CIP-0112 token standard V2: Expanded for institutional market structure

Strategic business capabilities

- **Multi-Prime Broker account segregation:** native (owner, provider, id) account hierarchies let clients use a single wallet across multiple Prime Brokers, each seeing only their managed accounts
- **Capital & liquidity efficiency:** built-in netting and transaction batching lower capital requirements and maximize operational throughput
- **Turnkey regulatory safeguards:** global asset pause controls give compliance teams immediate, system-wide risk management

Zero migration risk

- **Fully backwards-compatible:** runs seamlessly alongside V1 infrastructure with zero breaking changes for existing wallets, custody APIs, or DvP settlement engines.

SECTION 1B

Connecting to Canton: Deployment considerations at a glance

Deployment considerations: balancing your requirements

Choose the right balance of speed, data sovereignty, and architectural control — from a turnkey wallet solution, to a fully custom deployment.

Setup	Wallet Provider Turnkey Setup	Data Sovereign Wallet Setup	Fully Custom Wallet Setup
Choose when...	Speed matters most. You need fully managed solutions and upgrades.	Data/infrastructure control matters most. Want an institutional wallet but with full control over transaction data.	Architecture flexibility matters most. Need complete control.
Setup time	Days	Weeks	Months
Control	Standard enterprise grade control	High independent control	Maximum sovereignty and customization

[Jump to section 1C for details of available services and solutions](#) ready to support you, depending on how you choose to connect...

Deployment considerations: From quickstart to custom setup

Canton provides flexibility with modular setup to balance fast time to market, with long term infrastructure control.

This table shows you what components are needed to connect, and how access/deployment of these changes depending on your chosen model.

You are never locked into a single model: Start with a turnkey wallet and later integrate Bring-Your-Own-Validator or Wallet Gateway for data sovereignty, custom integrations, or specific DApp composability

Convenience / time to market → Customization over time

Setup		Wallet Provider Turnkey Setup	Data Sovereign Wallet Setup	Fully Custom Wallet Setup
Wallet components provided by:	**Wallet UI**	One stop shop from wallet provider: provides, manages, connects and integrates all Wallet components in a unified user experience out of the box: – Validator node – KMS – Policy engine – AML monitoring and reporting	Wallet UI from wallet provider that offers **Bring-Your-Own-Validator** Service	Select or create your own custom wallet UI, typically self-hosted
	Validator node		Bring your own validator (self-hosted or managed by a Node-as-a-Service provider) connected to the wallet provider's infrastructure	Self-hosted or managed by a Node-as-a-Service provider
	KMS		KMS provided through Wallet provider	Wallet provider or KMS provider (cloud or self-hosted)
	Institutional controls		Wallet provider offers all services (policy engine, AML monitoring & reporting etc.)	Self-hosted and custom built policy engine; custom built / 3rd party AML monitoring & reporting
	Integration		Use wallet provider's Bring-Your-Own-Validator Service to connect your own Validator Node — benefit from all other existing integrations	Use open source Wallet Gateway to orchestrate communication between components, or build your own custom integration

For clients of Asset Originators (e.g., DTCC): Setup checklist (1/2)

This is an initial guide to the building blocks needed. Please see the readiness checklist provided by the Asset Originator (e.g., DTCC) you are working with for full requirements

Entity	Required by:	Building block	Rationale
Asset Originator Client (e.g., Prime Broker)	**Canton**	CIP-0103[1], -0056, -0112 compliant wallet	Ensures client e.g. a Prime Broker - can securely authenticate, visualize, and authorize asset transfers while strictly enforcing internal risk and compliance policies
		Dars installed on wallet service's validator: – Asset Originator's Tokenization DApp dar (e.g. DTCC) – Cash Originator dar (e.g., Circle) – Trading Venue dar[1] (e.g., Tradeweb)	Enables Prime Broker's wallet service's Validator node (a wallet component) to validate both the tokenized asset (e.g. tUST) by the Asset Originator (e.g. DTCC) and the payment leg (e.g., USDC) by the Cash Originator (e.g. Circle) required for secure DvP settlement by the Trading Venue (e.g. Tradeweb). Payment leg visibility needed for GAAP reporting by Prime Broker
		PartyID / Allow-list Registry	Allows the Prime Broker to cryptographically prove eligibility to hold the restricted assets, verify counterparty compliance prior to executing a transfer. This info is consumed by Prime Broker via its wallet, and modifying allowlists Prime Broker can modify allowlist via its wallet and the Asset Originator's DApp
		Indicated as Observer in Trading Venue DApp	Prime Brokers are required by GAAP accounting practices to report both legs of a trade
		Vetting process for dars received from other entities	Streamlined code audits, issuer signature verification, internal approval processes before deploying third-party logic onto their secure nodes.

1 Only applies if the trading venue has a DApp. In future, for some trading venues Canton Network Utility and Token Standard V2 might suffice.

For clients of Asset Originators (e.g., DTCC): setup checklist (2/2)

Entity	Required by:	Building block	Rationale
Prime Broker (e.g., Marex)	**e.g. Off chain / IT integration**	If required: Web-2 IAM credentials to access Asset Originator DApp UI	Provides the guarding of Asset Originator UI to only be accessed by authorized DTCC members
		If required: Asset Originator segregated account (e.g., DTCC FBO account)	Provides the foundational legal structure to hold the underlying TradFi assets securely, bankruptcy-remote, and fully segregated
		Client operations dashboard enabled for digital assets (or API)	Upgrades the legacy TradFi interface, allowing client operations teams to seamlessly manage digital asset workflows, reporting, and lifecycle
		KYC/AML screening & UTXO monitoring	Satisfies regulatory requirements

Testing: please contact your Asset Originator for access and onboarding to their test environment and test plans.

For End Investors: setup checklist

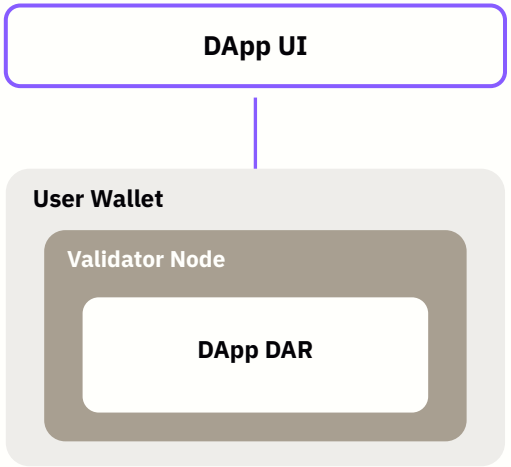
If using their own wallet, not the Prime Broker's

Entity	Required by:	Building block	Rationale
End Investor (e.g., DRW)	**Canton**	CIP-0103[1], -0056, -0112 compliant wallet	Ensures the trader can securely sign transactions, visualize complex multi-party settlements (DvP), and enforce internal algorithmic or manual risk limits prior to execution
		Dars installed on wallet service's validator: – Asset Originator Tokenization DApp dar (e.g. DTCC) – Cash Originator dar (e.g., Circle) – Trading Venue dar[1] (e.g., Tradeweb) Operator of validator to obtain dars and sign NDA/contract with dar provider as needed	Enables the trader's validator node (a wallet component) to validate both the tokenized asset (e.g., tUST) by the Asset Originator (e.g. DTCC) and the payment leg (e.g., USDC) by the Cash Originator (e.g. Circle) required for secure DvP settlement by the Trading Venue
		Vetting process for dars received from other entities	Streamlined code audits, issuer signature verification, internal approval processes before deploying third-party logic onto their secure nodes
	e.g. Off chain / IT integration	Digital asset account (e.g. tUST, USDC) with prime broker who is a member of the Asset Originator (e.g. DTCC)	Establishes the necessary offchain legal and custodial relationship required to compliantly hold and clear the tokenized asset
		Trading Venue (e.g. Tradeweb) account, e.g., for tUST:USDCx trading (and other pairs depending on trade)	Provides off chain onboarding and access to the secondary market liquidity pools and execution interfaces
		Wallet integrated into trading, risk management and accounting platform	Synchronizes onchain settlement events with the firm's internal off-chain ledgers, enabling PnL tracking and position management
		Legal agreements with counterparties	Ensures all onchain settlements are backed by binding off-chain legal frameworks (e.g., ISDA Master Agreements) for dispute resolution

1 Only applies if the trading venue has a DApp. In future, for some trading venues Canton Network Utility and Token Standard V2 might suffice.

How to connect and transact with a DApp

Interact with DApps through standard workflows enabling your organization to securely connect a Wallet to a DApp and access and transact across DApps on the Canton Network



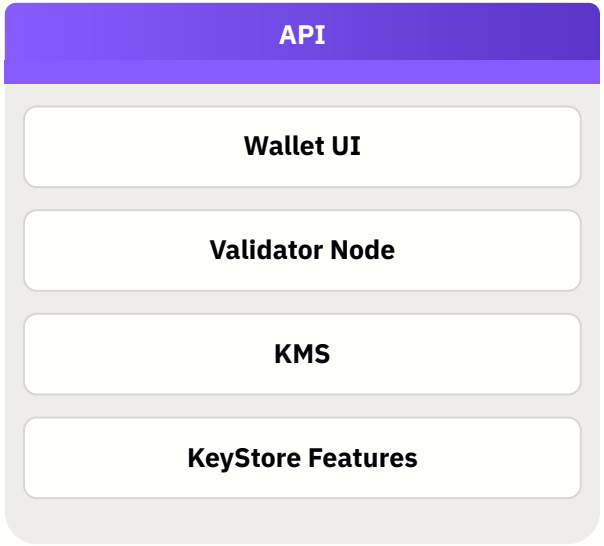
To engage with a DApp, the following sequence occurs:

- **Application Deployment (Node Readiness):** the DApp's specific business logic (DAR file) must be installed on your Validator Node. Note: for organizations using managed services, your Wallet or Node Provider handles this technical deployment on your behalf
- **Secure Authentication (Establishing Connection):** the application's interface (DApp UI) must securely link to your organization's wallet. Depending on your IT architecture, this is handled through a simple web-based “connect wallet” button or integrated directly via standardized APIs
- **Workflow Execution & Cryptographic Approval:** there is a strict separation of initiation and authorization. The user initiates the business workflow within the external DApp UI (or through an API), but provides the final, secure cryptographic signoff within their own controlled Wallet UI (or API) to execute the transaction

SECTION 1C

Solutions and services available to get you started

Turnkey wallet providers



Core capabilities:

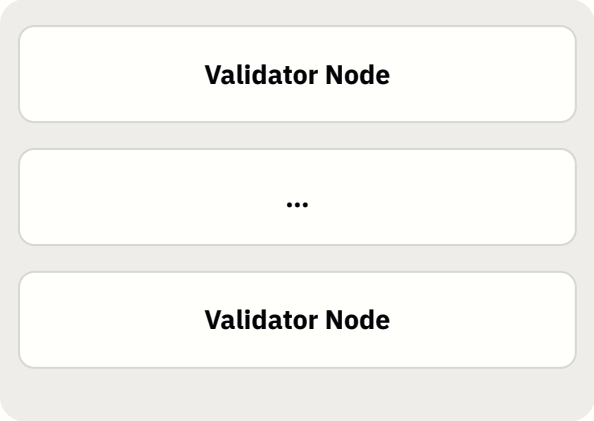
- Offer a Wallet Service as **fully integrated** solution off-the-shelf
- Builds or sources all required components
  - e.g. it might connect to a Node-as-a-Service (NaaS) Provider — detailed next
  - It might provide a dedicated Validator Node to each client, or agree to share one

Institutional wallets that support Canton today:

- BitGo
- Fireblocks
- Blockdaemon
- DFNS
- Kaleido
- Taurus

Jump to Section 3 to explore more customized setup models, and related services such as Bring Your Own Validator (BYOV), or Wallet Gateway Signing Drivers.

Node-as-a-Service (NaaS) providers

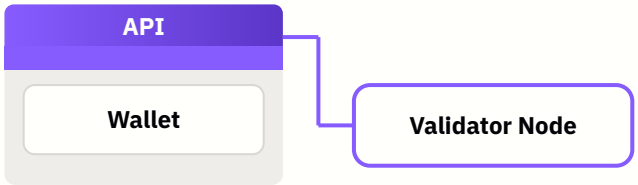


A NaaS Provider runs Validator Nodes as a **managed service** for clients and ensures:

- Uptime and data backups
- Compliance with protocol upgrades
- Installation, vetting, updating of DARs
- Wallet connectivity

See all NaaS providers that support Canton here

Wallet providers with Bring-Your-Own-Validator Node (BYOV) service

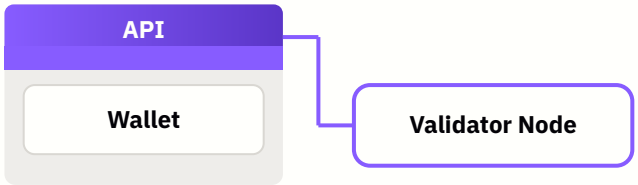


Wallet Providers may offer to **connect third party Validator Nodes** to their infrastructure, as an addition or replacement of their own Validator Node.

A user could connect their self-hosted/NaaS Validator Node to:

1. host the parties
2. read / write data

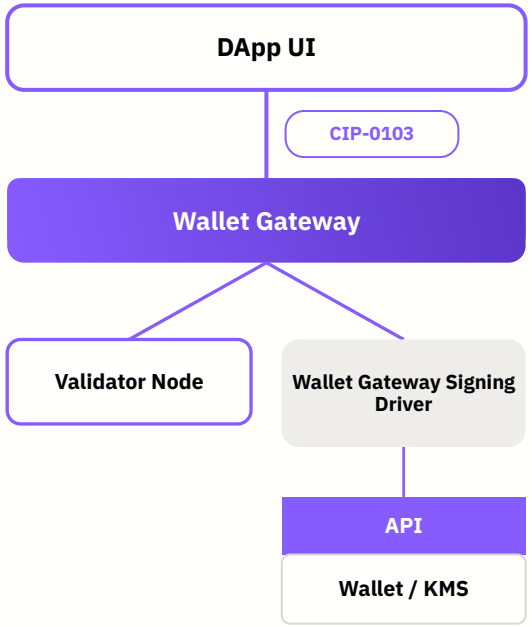
Example:
How to achieve data-sovereignty using BYOV



An institution might use a **Wallet Provider and it's BYOV Service** to connect their own validator node, and, e.g.:

- Source the Wallet from Wallet providers that offer a BYOV Service like Fireblocks/BitGo/Blockdaemon
- Source Validator node from a NaaS provider like Blockdaemon/IntellectEU/Kaleido or host your own Validator
- Connect the Validator to the Wallet Provider using the BYOV Service

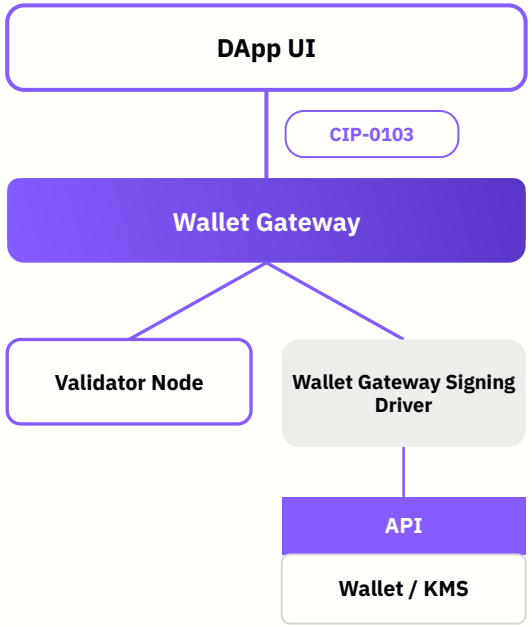
Wallet Gateway



An open source, free to use component by Canton Foundation that allows users to connect to any:

- CIP-0103 compliant **DApp UI** — even if user's wallet does not support CIP-0103
- **Wallet provider (or self-hosted KMS)** that provides a Wallet Gateway Signing Driver
- Self-hosted/NaaS **Validator Node**, in addition to, or to replace the Wallet Provider's Validator Node

Example: Build a custom wallet using Wallet Gateway



Any component of a Wallet can be sourced or self-hosted instead of using a turnkey wallet

An institution might build a custom wallet, leveraging **Wallet Gateway to orchestrate components** for their custom wallet, and, for example:

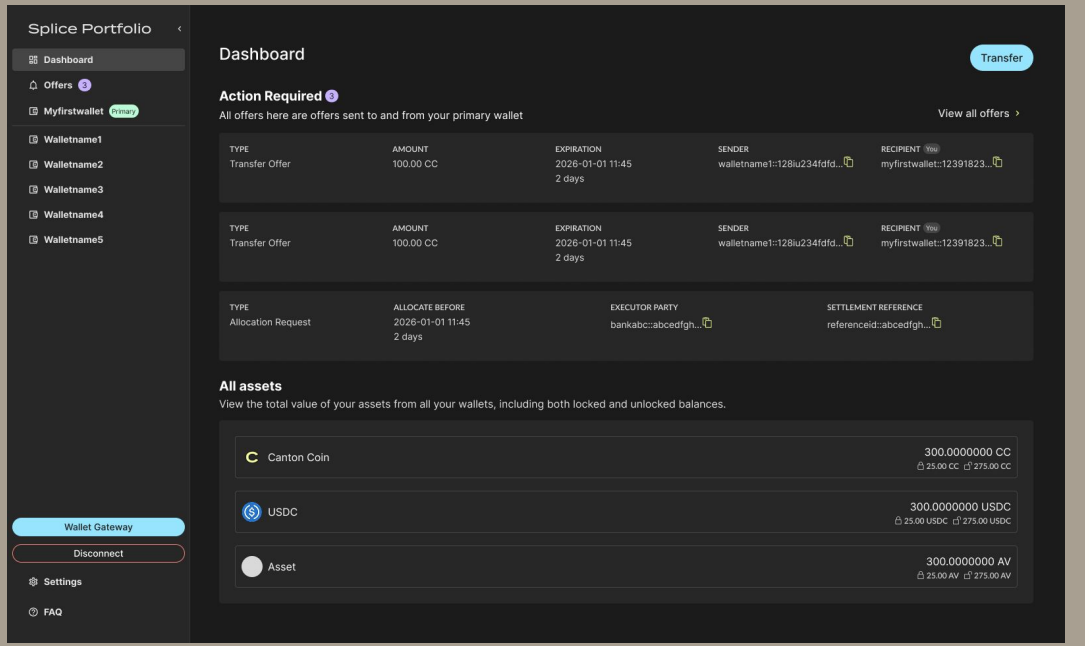
- Source a KMS from a hot and cold KMS provider (e.g. MPCH), cloud KMS provider (e.g. AWS, Azure), a wallet provider with KMS capabilities (e.g. Fireblocks, BitGo, Blockdaemon) or self host via HSM infra (e.g. MPCH, Securosys, Thales Luna). See note below.
- Source a Validator node from a NaaS provider or host your own Validator.
- Build/buy own Wallet UI or directly integrate into trading systems UI.

Note: for each solution, a Wallet Gateway signing driver is required. Many wallet providers have already published theirs, and signing drivers for several KMS systems are available.

Use BYOV for sovereign data control & Wallet Gateway for maximum architectural flexibility

	BYOV (Bring Your Own Validator Node)	Wallet Gateway
Description	A commercial service offered by Wallet Providers to connect a user's (or third party's) Validator Nodes to the Wallet Provider infrastructure.	A free, open-source component provided by Canton Foundation to connect a Wallet/KMS, DApp UI, and Validator Node together
Goal	User's **Data Sovereignty**: full control of Validator Node	User's **Flexibility**: option to combine and connect different components across separate trust zones. This can include user's **Data Sovereignty** by having full control of Validator Node
DApp connectivity	Dependent on Wallet Provider's CIP-0103 compliance	Built-in CIP-0103 compliance
Node access	Connects user's Self-Hosted or NaaS Validator Node to the Wallet Provider’s infrastructure (as an addition or replacement)	Orchestrates user's self-hosted or NaaS Validator Node communication with a DApp UI and signing service
KMS access	Relies on the Wallet Provider's integrated KMS	Connects to any Wallet Provider or self-hosted KMS that features a Wallet Gateway Signing Driver

DApp Portfolio UI



An open source, free to use Token Standard compliant component that allows users to:

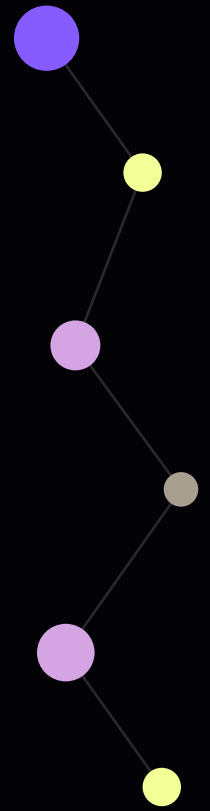
- **View and interact** with their assets/positions in Canton
- **Perform advanced workflows** such as offers, allocations, and multi-party views

- It can offer this functionality in case Wallets do not offer this natively
- The DApp is **CIP-0103 compliant**, and supports CIP-0103 compliant Wallets and the Wallet Gateway

Canton

Section 2

Setup considerations: DAR, Node and DApp dependencies across wallet options



DAR dependencies for all wallet setup options

Asset Originator (e.g., DTCC) client transactions to be triggered by the client in the Asset Originator's DApp UI

Node	Wallet / KMS	Dar required by	Note
Turnkey Wallet Provider's	Wallet provider	Wallet Provider	
Self-hosted	Self-hosted KMS (internal parties)[2]	Asset Originator Client	
	Self-hosted KMS (external parties)	Asset Originator Client	
	Wallet provider	Asset Originator Client	Asset Originator Client needs dar even if Wallet Provider has dar, as Asset Originator Client Node is used instead of Wallet Provider Node
NaaS	NaaS-hosted KMS (internal parties)	NA[3]	
	NaaS- or self-hosted KMS (external parties)	NaaS	
	Wallet provider (using NaaS and not Wallet Provider's own node)	NaaS	NaaS needs dar even if Wallet Provider has dar, as NaaS Node is used instead of Wallet Provider Node

Wallet to Node connectivity options for all wallet setup options

Node	Wallet / KMS	Wallet to node connectivity options
Turnkey Wallet Provider's	Wallet provider	Wallet Gateway[4] Wallet Provider integration
Self-hosted	Self-hosted KMS (internal parties)[2]	Direct connection of KMS to node
	Self-hosted KMS (external parties)	Wallet Gateway[4] Custom integration
	Wallet provider	Wallet Gateway[4] Bring-Your-Own-Validator
NaaS	NaaS-hosted KMS (internal parties)	NA[3]
	NaaS- or self-hosted KMS (external parties)	Wallet Gateway[4] Custom integration
	Wallet provider (using NaaS and not Wallet Provider's own node)	Wallet Gateway[4] Bring-Your-Own-Validator

2 Internal parties are not recommended for production volume — external parties combined with separate trust zones (e.g. via Wallet Gateway) ensure a compromised validator cannot send transaction requests directly to the HSM.

3 Only listed for completeness — not recommended to use a NaaS with an internal party for security reasons.

4 While in most cases there are alternatives, Wallet Gateway is the preferred choice.

Where to Trigger Token-Standard Transactions (e.g. Transfer, Allocation, etc.) for all wallet setup options

Node	Wallet / KMS	Recommendation on UI to trigger transactions
Turnkey Wallet Provider's	Wallet provider	Wallet UI
Self-hosted	Self-hosted KMS (internal parties)[2]	Custom, e.g., command line
	Self-hosted KMS (external parties)	Custom (e.g. command line), or Wallet Gateway with End-Investor hosted Portfolio UI, Trading Venue onboarding UI, Asset Originator onboarding DApp UI
	Wallet provider	Wallet UI
NaaS	NaaS-hosted KMS (internal parties)	NA[1]
	NaaS- or self-hosted KMS (external parties)	Wallet Gateway with End-Investor hosted Portfolio UI, Trading Venue onboarding UI, Asset Originator onboarding UI
	Wallet provider (using NaaS and not Wallet Provider's own node)	Wallet UI

Wallet Gateway can be self-hosted by the institution or hosted by a third party such as the node provider or wallet provider.
1 It is not recommended to use a NaaS with an internal party.

Wallet to DApp connectivity options (to trigger non-Token-Standard transactions) for all wallet setup options

E.g., custom Asset Originator client (e.g. DTCC Client) transactions might be triggered by client in the Asset Originator DApp UI

Node	Wallet / KMS	Wallet to DApp UI connectivity options
Turnkey Wallet Provider's	Wallet provider	
Self-hosted	Self-hosted KMS (internal parties)[2]	
	Self-hosted KMS (external parties)	
	Wallet provider	Wallet Gateway[4] CIP-0103 standard support
NaaS	NaaS-hosted KMS (internal parties)[3]	
	NaaS- or self-hosted KMS (external parties)	
	Wallet provider (using NaaS and not Wallet Provider's own node)	

2 Internal parties are not recommended for production volume — external parties combined with separate trust zones (e.g. via Wallet Gateway) ensure a compromised validator cannot send transaction requests directly to the HSM.

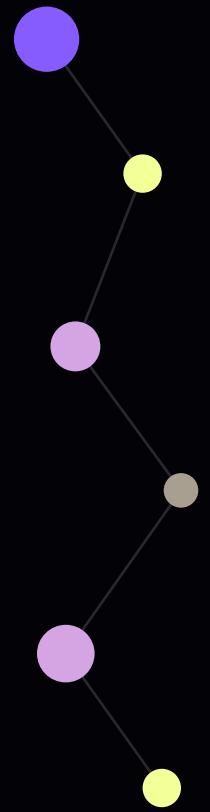
3 Only listed for completeness — not recommended to use a NaaS with an internal party for security reasons.

4 While in most cases there are alternatives, Wallet Gateway is the preferred choice.

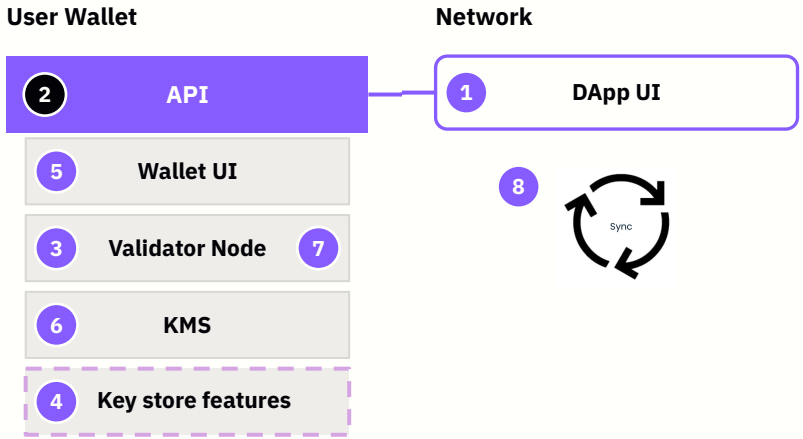
Canton

Section 3

Canton Network information flow based on chosen setup option



Information flow across components: A turnkey wallet provider's Wallet directly connected to DApp UI

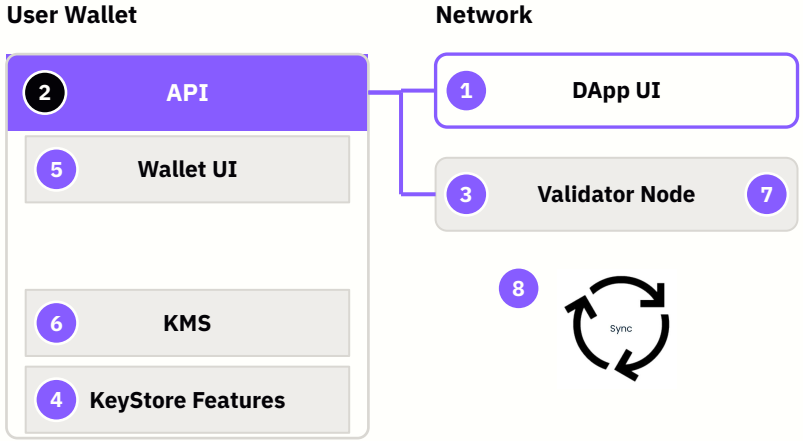


If the transaction is initiated in the Wallet UI (Token Standard functionality only): steps 1–2 are replaced — the user initiates the transaction in the Wallet UI and the Wallet generates the transaction request for processing.

Information and user action flow during a DApp transaction if the transaction is initiated in DApp UI:

1. **DApp UI:** user connects their wallet, initiates transaction
2. **Wallet API:** receives the generated transaction command from the DApp UI for processing
3. **Validator Node (Preparation):** prepares execution payload and computes the cryptographic hash for signing
4. **KeyStore Features:** analyzes the transaction against compliance and business rules (e.g., AML, multi-admin policy engine approvals)
5. **Wallet UI:** user reviews the parsed transaction details and grants final approval
6. **KMS:** securely signs the cryptographic hash with the user's private key
7. **Validator Node (Submission):** encrypts the payload and signed hash, submitting it to the network/synchronizer with the relevant PartyIDs
8. **Synchronizer:** routes the encrypted message to the Validator Nodes hosting the involved PartyIDs

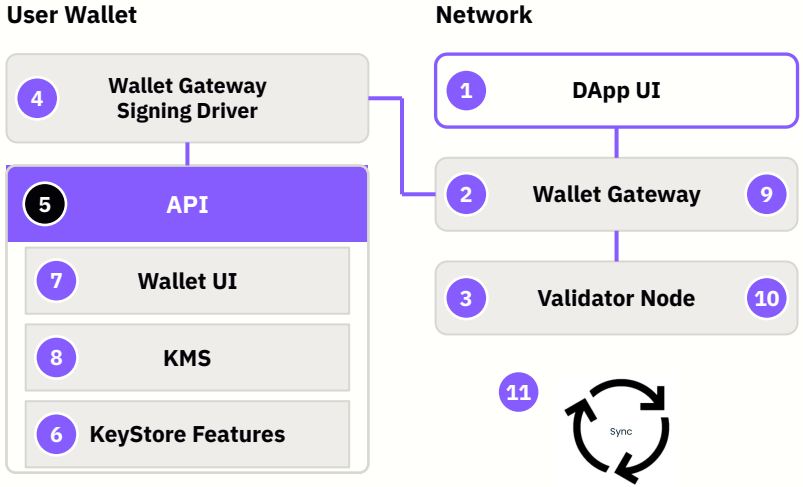
Information flow across components: A Wallet directly connected to DApp UI, using BYOV



The workflow is identical to using a regular turnkey wallet provider's Wallet, except **Validator Node functions (steps 3 & 7) execute externally** via Wallet API routing.

The external Validator Node is connected to the Wallet, enabled by the Wallet Provider's **Bring-Your-Own-Validator Node Service (BYOV)**.

Information flow across components: Turnkey wallet connected to external Validator & DApp UI via Wallet Gateway

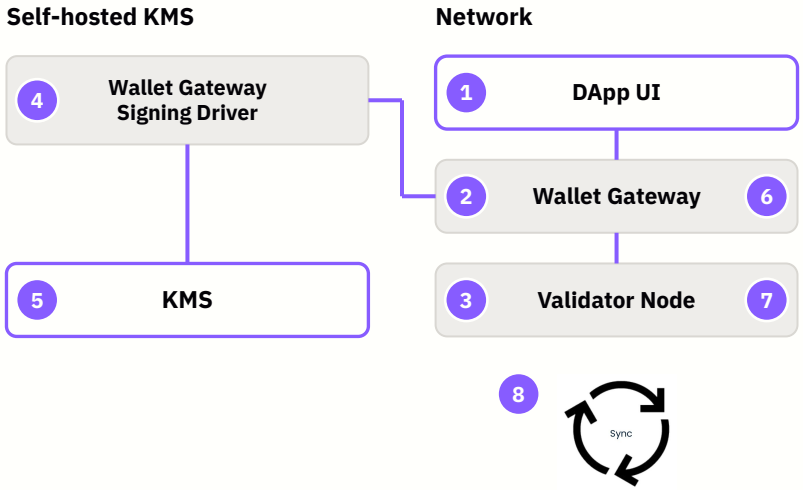


Information and user action flow during a DApp transaction if the transaction is initiated in DApp UI:

1. **DApp UI:** user connects their Wallet Gateway, initiates transaction
2. **Wallet Gateway (Preparation):** receives the generated transaction request from the DApp UI for processing
3. **Validator Node (Preparation):** prepares execution payload and computes the cryptographic hash for signing
4. **Wallet Gateway Signing Driver:** presents hash and information in a format convenient for the Wallet API
5. **Wallet API:** receives the Validator Node-prepared transaction command from the Wallet Gateway Signing Driver for processing
6. **KeyStore Features:** analyzes the transaction against compliance and business rules (e.g., AML, multi-admin policy engine approvals)
7. **Wallet UI:** user reviews the parsed transaction details and grants final approval
8. **KMS:** securely signs the cryptographic hash with the user's private key
9. **Wallet Gateway (Submission):** receives the signed hash from the Wallet (via API / Signing Driver) and adds the previously prepared transaction payload

Steps 10 and 11. The same as without Wallet Gateway, except the user's (or a third party's) Validator Node is used for submission of the data to the network.

Information flow across components: KMS connected to DApp UI via Wallet Gateway



Information and user action flow during a DApp transaction if the transaction is initiated in DApp UI:

1. **DApp UI:** user connects their Wallet Gateway, initiates transaction
2. **Wallet Gateway (Preparation):** receives the generated transaction request from the DApp UI for processing
3. **Validator Node (Preparation):** prepares execution payload and computes the cryptographic hash for signing
4. **Wallet Gateway Signing Driver:** presents hash and information in a format convenient for the KMS
5. **KMS:** securely signs the cryptographic hash with the user's private key
6. **Wallet Gateway (Submission):** receives the signed hash from the Wallet (via API / Signing Driver) and adds the previously prepared transaction payload

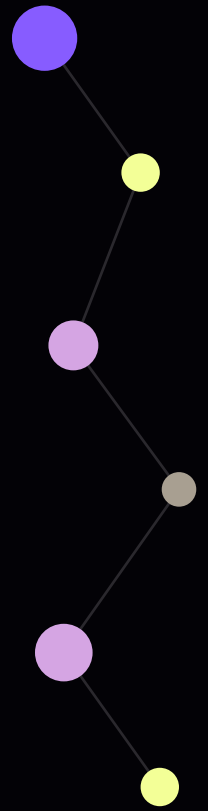
Steps 7 and 8: same as without Wallet Gateway, except the user's (or a third party's) Validator Node is used for submission of the data to the network.

Note: institutions can use the Wallet Gateway in combination with a KMS to build their own wallet. Key-Store Features and UIs can be added as needed.

Canton

Section 4

Canton Network DApp best practices



DApps must provide a DApp UI to ensure scalability

	Wallet UI: standard functionality	DApp UI: custom functionality
Best suited for	• Treasury Management • Token Standards (e.g. CIP-0056, CIP-0112) • Transfers • Allocations • Delegation	• Advanced functionality, e.g. bespoke logic, regulatory compliance • Specific use case examples: – Allow-list management – Trading Venue onboarding – DEX order submission
Benefit	**Focus on maximum security,** human-readable transaction visualization and workflows of standard transactions shared across DApps	**Focus on DApp-specific user experience, customization, and delivery speed.** DApp UIs can include Token Standard functionality for better user experience