The below document outlines a detailed design for building a FinP2P compatible digital securities solution over the Tezos blockchain. This design document focuses on the Asset Issuance use case.

Introduction **1**

Scope **3**

Asset Issuance 3

High Level Implementation **3**

Asset Issuance 3

Detailed Design **4**

The DLT Adapter 6

Key Management 9

Communicating with the Tezos node 10

Tezos Smart Contracts 10

The token functionality 10

FinP2P Signature validation contracts 12

Introduction

FinP2P (https://finp2p.atlassian.net/wiki/spaces/FINP2P/overview) aims to digitize, systematize and automate private capital markets, by enabling regulated financial institutions to connect to a global network of digital private securities. To this network, each institution can bring from one side verified assets (to register, manage and offer their securities), and bring from the other side verified users (to invest and manage their securities ownership online).

Ownera provides a FinP2P node, compatible with the FinP2P specifications, that provides a set of APIs as the entry point for interacting with the FinP2P Network and can be to build digital securities services on top of the FinP2P network, which are agnostic to the DLT used underneath the FinP2P node.

Tezos is a decentralized, open-source blockchain network that can execute peer-to-peer transactions and serve as a platform for deploying smart contracts. The native cryptocurrency for the Tezos blockchain is the tez which has the symbol XTZ. The Tezos network achieves consensus using a liquid proof-of-stake model. The primary protocol of Tezos utilizes liquid proof of stake (LPoS) and supports Turing-complete smart contracts in a domain-specific language called Michelson. Michelson is a purely functional stack-based language with a reduced instruction set and no side effects, designed with formal verification in mind. Tezos

features an on-chain governance model that allows the protocol to amend itself when upgrade proposals receive a favorable vote from the community.

This solution aims to bring the Tezos blockchain into the FinP2P network for financial organizations implementing digital securities.

Scope

Asset Issuance

This solution will include an Ownera end to end digital securities platform based on the Tezos blockchain, that allows financial institutions to issue and trade digital securities.

1. Implement a digital securities tokenization solution under the Tezos blockchain, which is compatible with the FinP2P specifications, and allows organizations to issue digital assets on the Tezos blockchain.
2. Through the use of the FinP2P node, those assets issued on the Tezos blockchain can be published to other FinP2P Nodes so that their users can invest in those assets, providing global liquidity for those assets.

High Level Implementation

Asset Issuance

1. Implementation of a set of Tezos smart contracts for tokenization of digital securities .
    a. Based on the **Tezos FA2 specifications** for multi-asset interface for tokens and multi-token contracts on Tezos
    b. Using a **transfer hook pattern** for implementation of the custom permissions required by the FinP2P specification, and the implementation of digital securities verifications.
    c. integrations into the Ownera Regulation compliance platform
    d. Implementation of a FinP2P transaction signature verification on the Tezos smart contracts.
2. Implementation of a FinP2P DLT adapter, that implements the DLT specification, and allows to interconnect the Tezos tokenization solution described in #1 to the FinP2P network .
    a. Allow managing FinP2P resources under the Tezos blockchain
    b. Allow digital securities tokenization transactions under the Tezos blockchain
3. Implementing helm charts and infrastructure solutions for a kubernetes based private network environment for the Tezos blockchain.

Detailed Design

The core functionality of FinP2P is to serve as an abstraction layer that connects multiple underlying ledger technologies, and allow both access to them with a unified API, as well as seamless connectivity between them. An intermediary, thin and stateless network layer, interconnecting between any app and any approved distributed ledger technology, to which all financial institutions can easily connect. Each financial institution connecting to the network will be described as a Node.

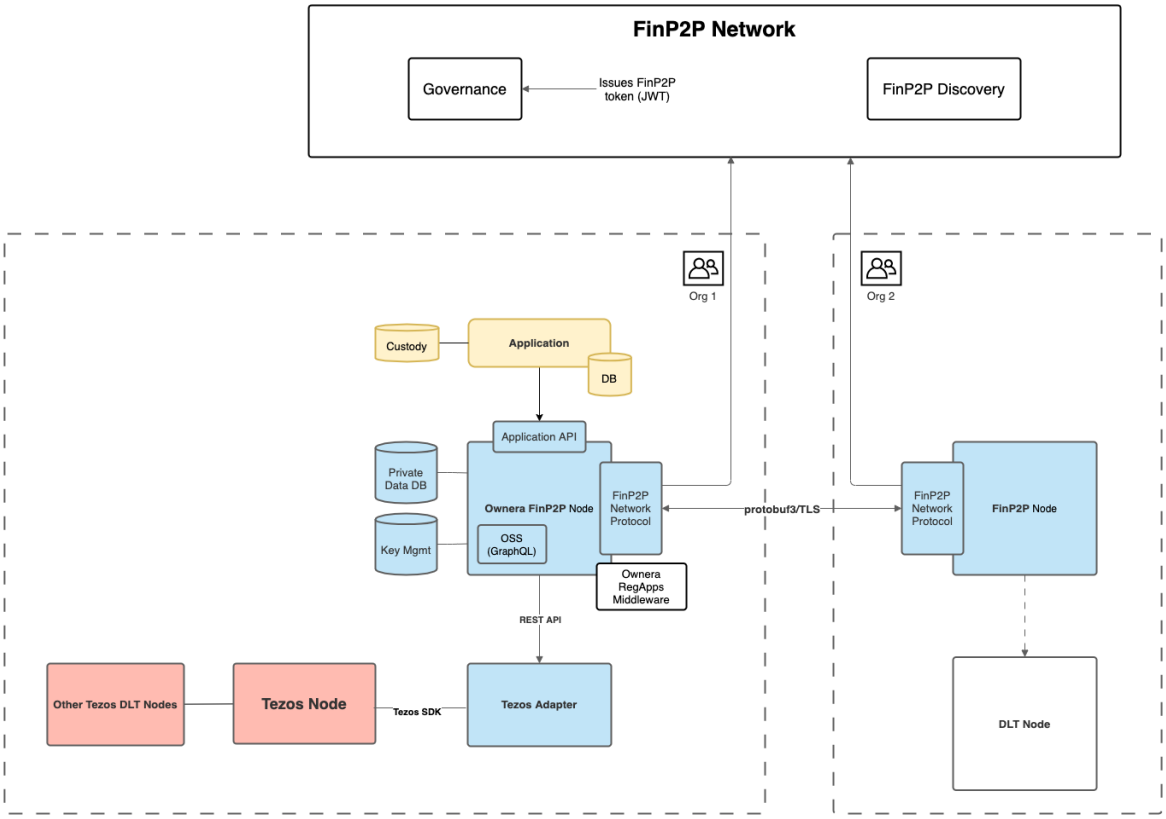
FinP2P defines a network protocol and a set of specification APIs that aim to provide a communication abstraction layer between applications, data privacy and distributed ledger implementations.

Ownera has developed a FinP2P Node that implements the FinP2P specifications, it exposes a set of APIs for applications and integrations.

The implementation of A Tezos Asset Issuance in FinP2P, involves an implementation of a FinP2P DLT Adapter that implements the functionality of executing transactions on an asset managed in an underlying platform. This adapter will allow other organizations to allow their clients to invest in the published asset.

An organization deploying Ownera FinP2P for Tezos, will run an Ownera FinP2P Node, connected to the FinP2P network, a Tezos node connected to the Tezos network and a new FinP2P DLT Adapter component which will connect the Tezos Node to FinP2P.
The DLT Adapter to be developed, will implement the FinP2P DLT Adapter interface from one hand, and will use the Tezos SDK to communicate with the Tezos node.

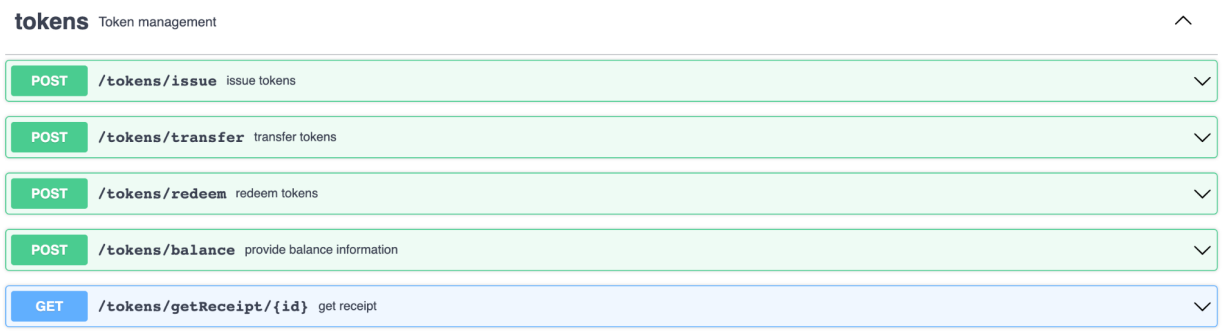
The diagram below illustrates the basic structure of a FinP2P network stack.



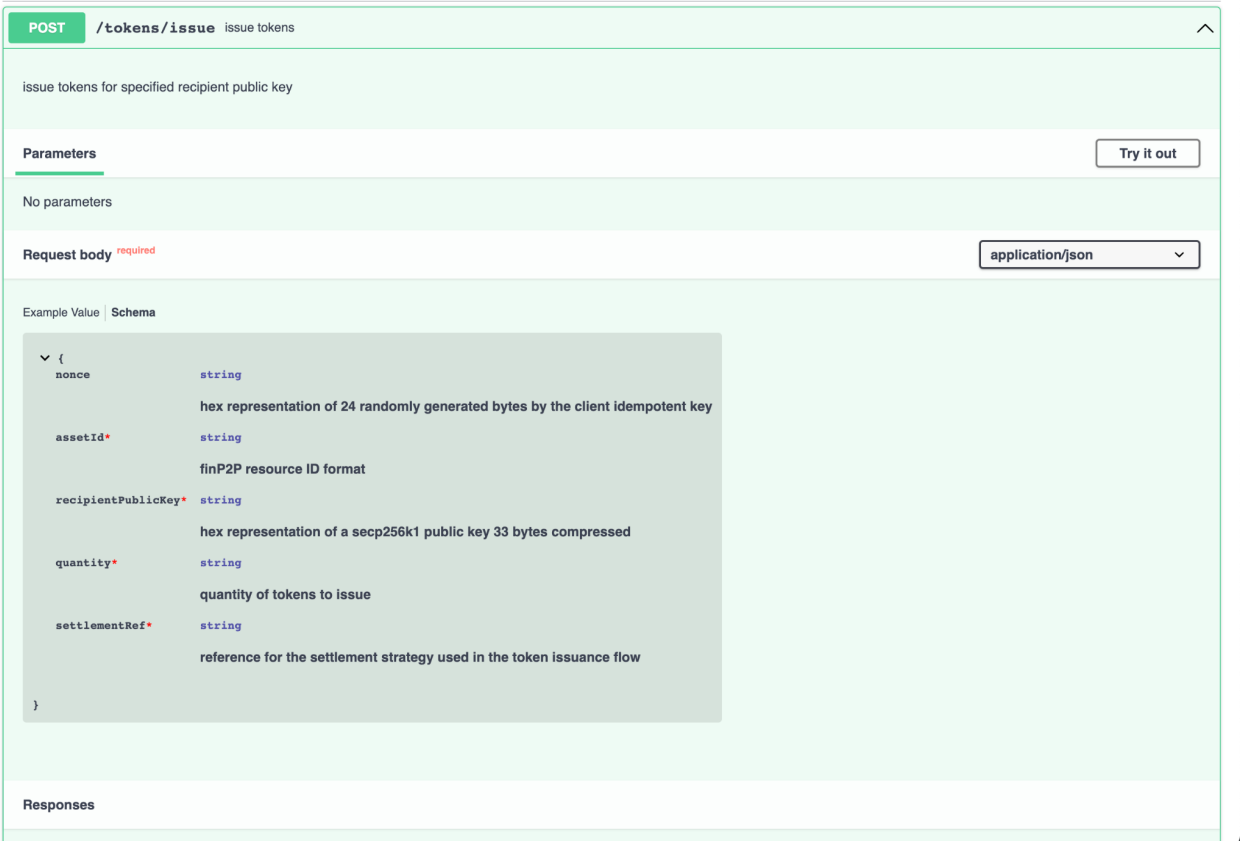
The DLT Adapter

The DLT Adapter API is a specification of the Restful APIs that should be implemented by a DLT Adapter under the Ownera FinP2P Node that abstracts the interaction between the Ownera FinP2P Node and the underlying Digital Securities Ledger.

The following APIs are to be implemented by the DLT Adapter



Details information about the API's to implement here:
https://finp2p-docs.ownera.io/reference/dlt-adapter-spec-1



POST `/tokens/transfer` transfer tokens

transfer tokens from sender to recipient

Parameters Try it out

No parameters

Request body required — application/json

Example Value | **Schema**

```
{
  nonce*                string   nonce
                                 32 bytes buffer (24 randomly generated bytes by the client + 8 bytes epoch timestamp seconds) encoded to hex
  assetId*              string
                                 the asset id
  sourcePublicKey*      string
                                 source owner hex representation of a secp256k1 public key 33 bytes compressed
  recipientPublicKey*   string
                                 recipient owner hex representation of a secp256k1 public key 33 bytes compressed
  quantity*             string
                                 the token quantity to transfer
  signatureTemplate*    signatureTemplate > {...}
  settlementRef*        string
                                 reference key for the settlement strategy used in the transfer flow
}
```

POST `/tokens/balance` provide balance information

provide token balance information

Parameters Try it out

No parameters

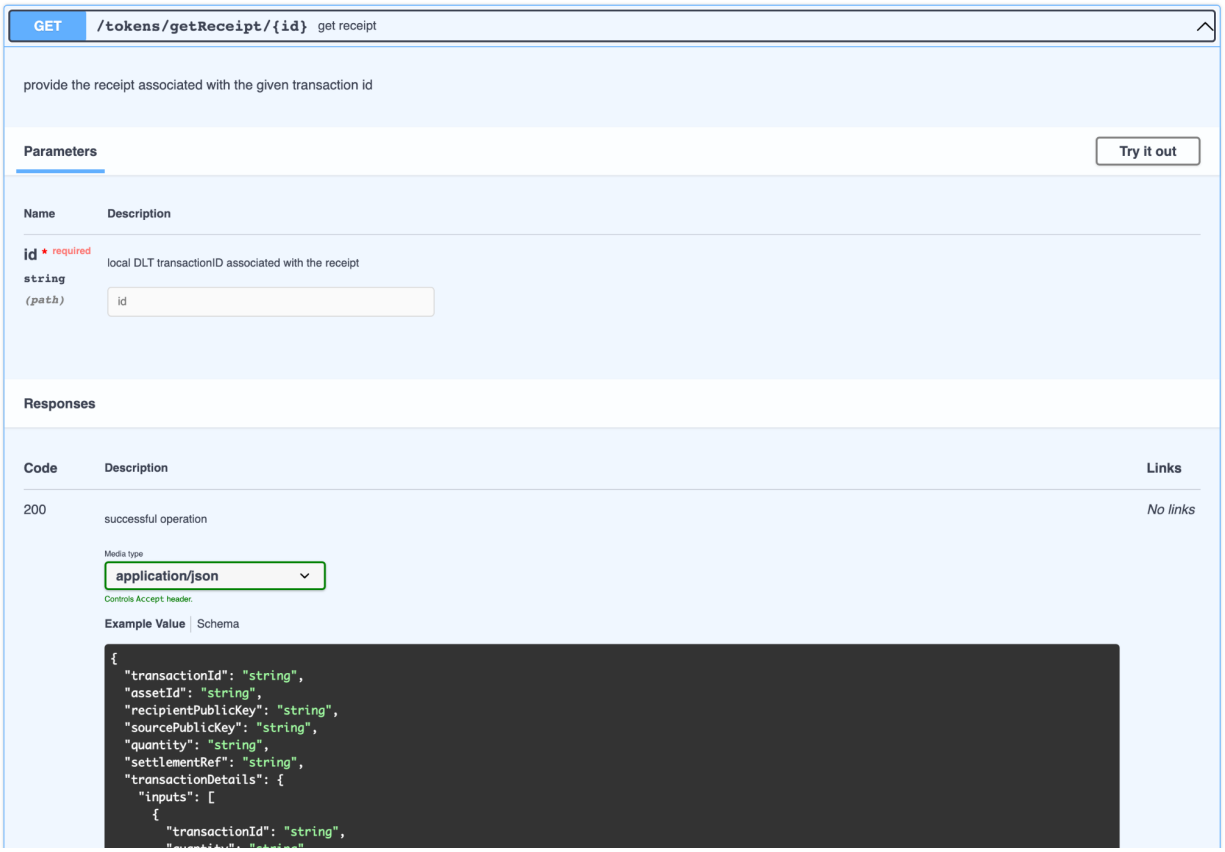
Request body required — application/json

Example Value | Schema

```
{
  "assetId": "string",
  "sourcePublicKey": "string"
}
```

Responses

Code	Description	Links
200	successful operation Media type: application/json Controls Accept header. **Example Value** | Schema `{ "quantity": "string" }`	*No links*



Key Management

The DLT adapter is responsible for maintaining a local Tezos specific cryptographic key that is mapped to the FinP2P ID of the user.
Every transaction signed by the user, will be signed by the user FinP2P key, the signature will be validated by the FinP2P node sitting next to the DLT adapter.
The DLT adapter should use the local keys of the user in order to sign transactions to the Tezos blockchain.
As an extra security measure, the Tokenization smart contracts on Tezos may verify the FinP2P signature by the user (in addition to the standard transaction validations by the Tezos blockchain). This ensures that the local keys by itself are not enough to perform actions on behalf of the user but requires the FinP2P signature as well.

In order for the DLT adapter to manage the local user keys, the Ownera Custody adapter can be used. The Custody adapter provides an abstraction on top of various industry solutions for custody and key management, it can create keys and sign transactions.

Communicating with the Tezos node

The DLT adapter will serve as a Tezos client, and will use one of the Tezos client libraries to communicate with the Tezos node
NodeJS - https://tezostaquito.io/
GoLang - https://github.com/blockwatch-cc/tzgo, https://github.com/goat-systems/go-tezos

The Tezos node GraphQL API can be used to query the ledger for receipt, balance or other needs.

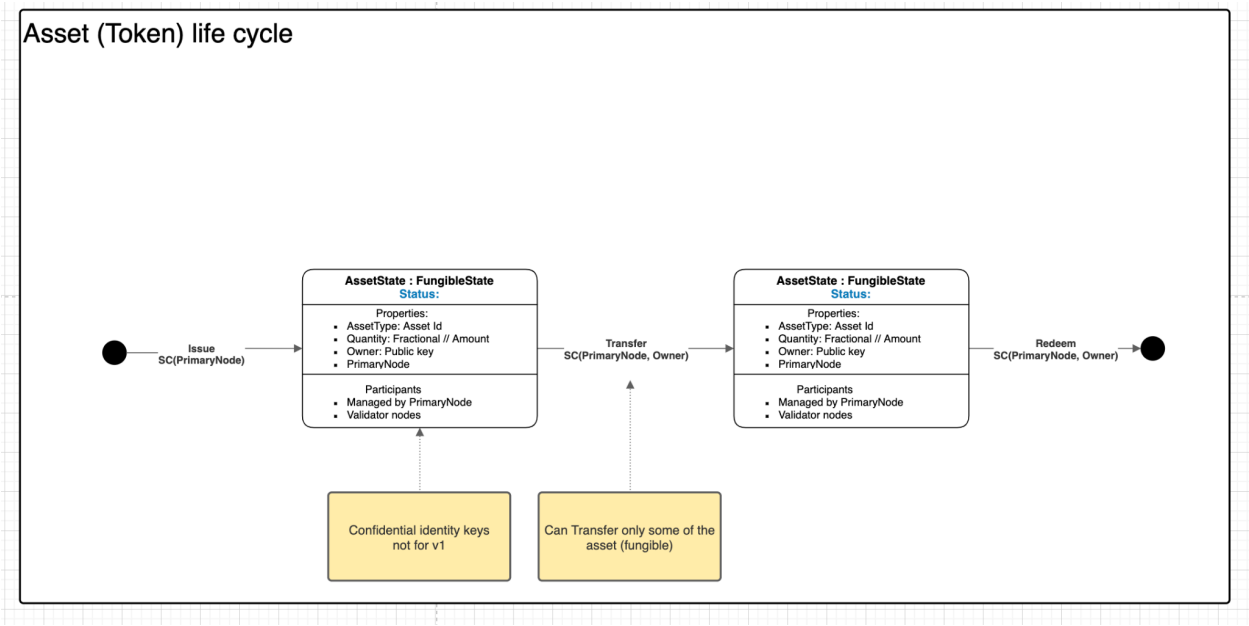
Tezos Smart Contracts

The token functionality

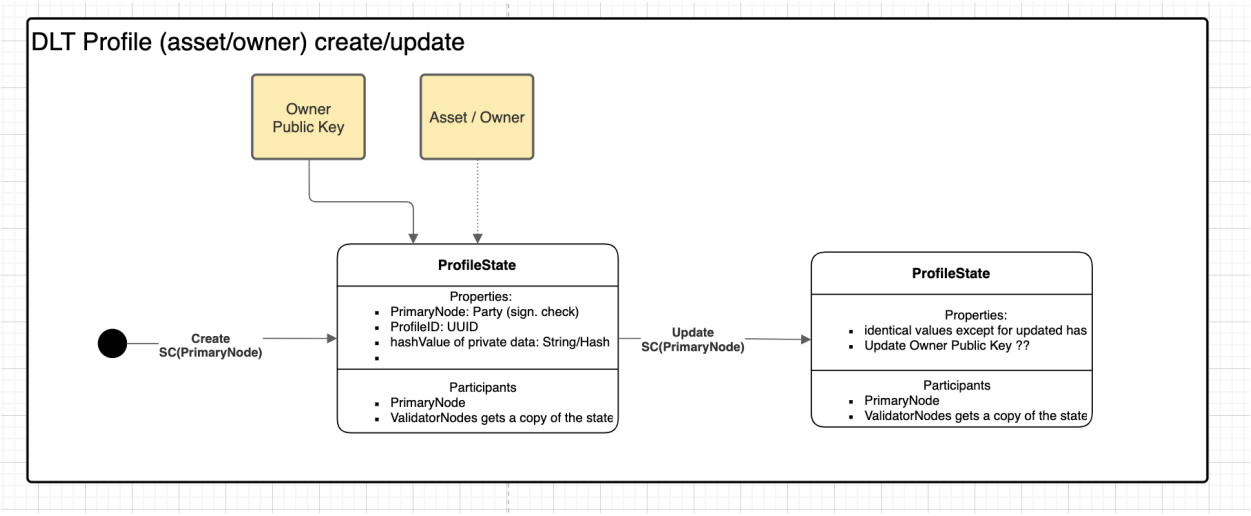
The tezos smart contract implementation needs to support the following use cases:

1) Minting tokens by the organization node to owners (Issue)
2) Transfer tokens between owners (Transfer)
3) Redeem tokens for an owners, the tokens will be “burned” (Redeem)
4) Get balance for owner on a specific asset (GetBalance)
5) Get receipt of a token transaction given a tezos transaction ID (GetReceipt)

The below diagram illustrates the State Transition of a typical Asset/Token



The below diagram illustrates the State Transition of a typical User/Owner DLT profile, currently the management of this profile can be deferred.



FA2 should be used for supporting a token transaction movement.

A reference implementation should be used for testing purposes, the following libraries can be considered:

FA2 - https://gitlab.com/smondet/fa2-smartpy/ (https://assets.tqtezos.com/docs/token-contracts/fa2/1-fa2-smartpy/)

FA2 is a token standard on Tezos. The standard is based on a proposal for a unified token contract interface. It addresses two aspects of importance to token standards: Token type and permission standardisation.

While FA1.2 requires multiple smart contracts for multiple tokens, FA2 allows single- and multi-token smart contracts for a variety of tokens
With FA2, those implementing the token contracts can configure freely:

- the token type(s),
- the token management (administration, whitelisting, etc.),
- the supply operations (minting and burning tokens),
- the permissioning architecture, and
- questions on contract upgradability.

Specifically the permissioning architecture relates to whether permissioning is determined in the contract, i.e. a monolith, with a transfer hook to another contract, or a separate wrapper contract. Each of this options represents an FA2 implementation pattern for permissions.

FinP2P Signature validation contracts

The movement of tokens may be governed by a FinP2P signature validation rule.
The smart contract may inspect the FinP2P signature (added as extra payload on the transaction) and verify it matches the owner's FinP2P public key and transaction information.

TBD using wrapper implementation vs Transfer hook for implementation of the signature validation contracts

In a wrapper implementation, the separate wrapper contract applies permissions by forwarding calls to the main contract, which manages the token ledger. Wrappers allow for modularity and extending the functionalities of the main token contract. Moreover, they allow for easier upgrades and replacements. Wrapper approaches can lead to fragmentation and increase complexity.

When using transfer hooks, the main token contract calls another contract, which includes the permissioning specifications, i.e. "permissioning policies". Its main benefits are a separation of concerns, granular permissioning rules, and the possibility to upgrade permissions. At the same time, transfer hooks bring gas limitation concerns, as do wrappers. Hooks are trickier security-wise and are much more gas expensive than wrappers. Additionally, permissioning policies are more complex and the contract size increases with contract use because of increasing permissioning policies.