Tokens in Hyperledger Fabric

What's possible today and what's coming

Elli Androulaki (lli@zurich.ibm.com)
Angelo De Caro (adc@zurich.ibm.com)
Kaoutar Elkhiyaoui (kao@zurich.ibm.com)
David Enyeart (enyeart@us.ibm.com)
IBM Research - Zurich

Goals

- Learn about the Initiatives around Tokens
- What’s possible today
- What’s coming

Agenda

01

Introduction

02

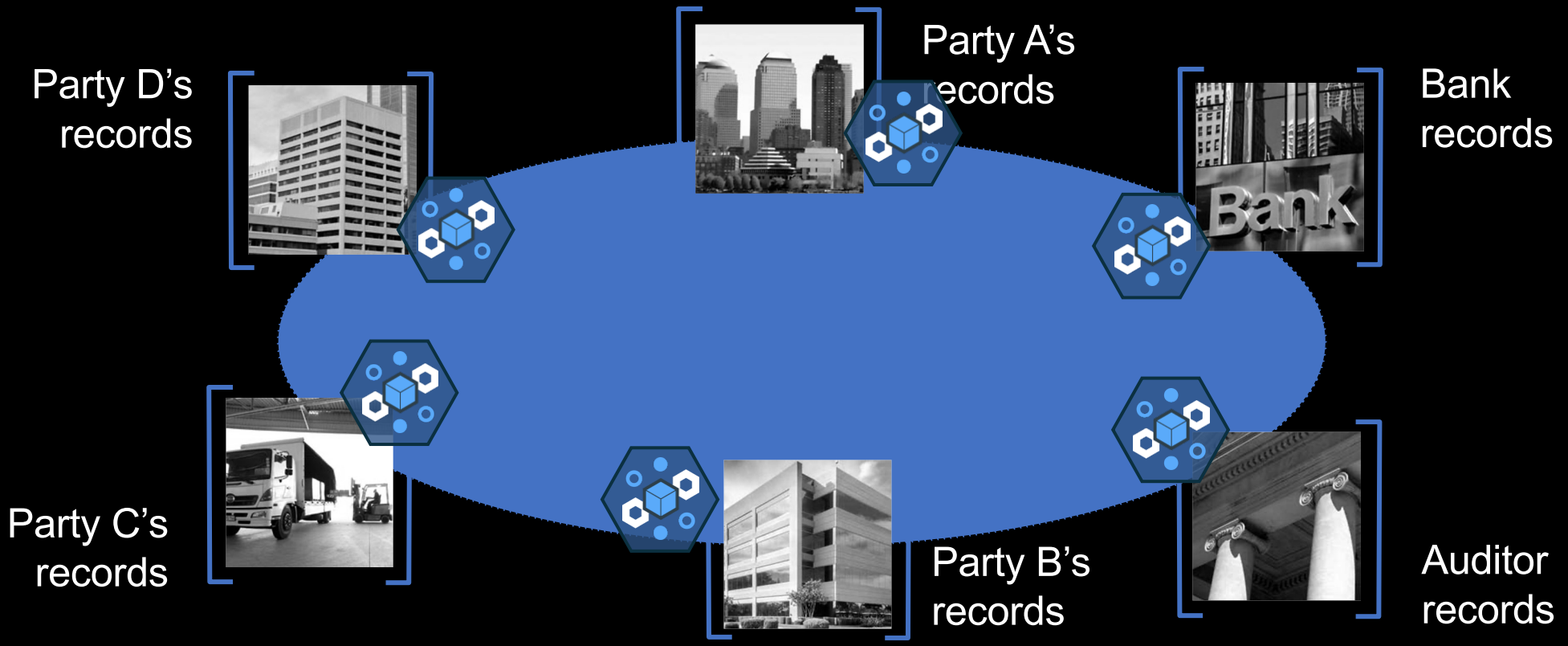
What's possible today?

03

What's coming? Fabric Token-SDK

Permissioned Blockchain Paradigm

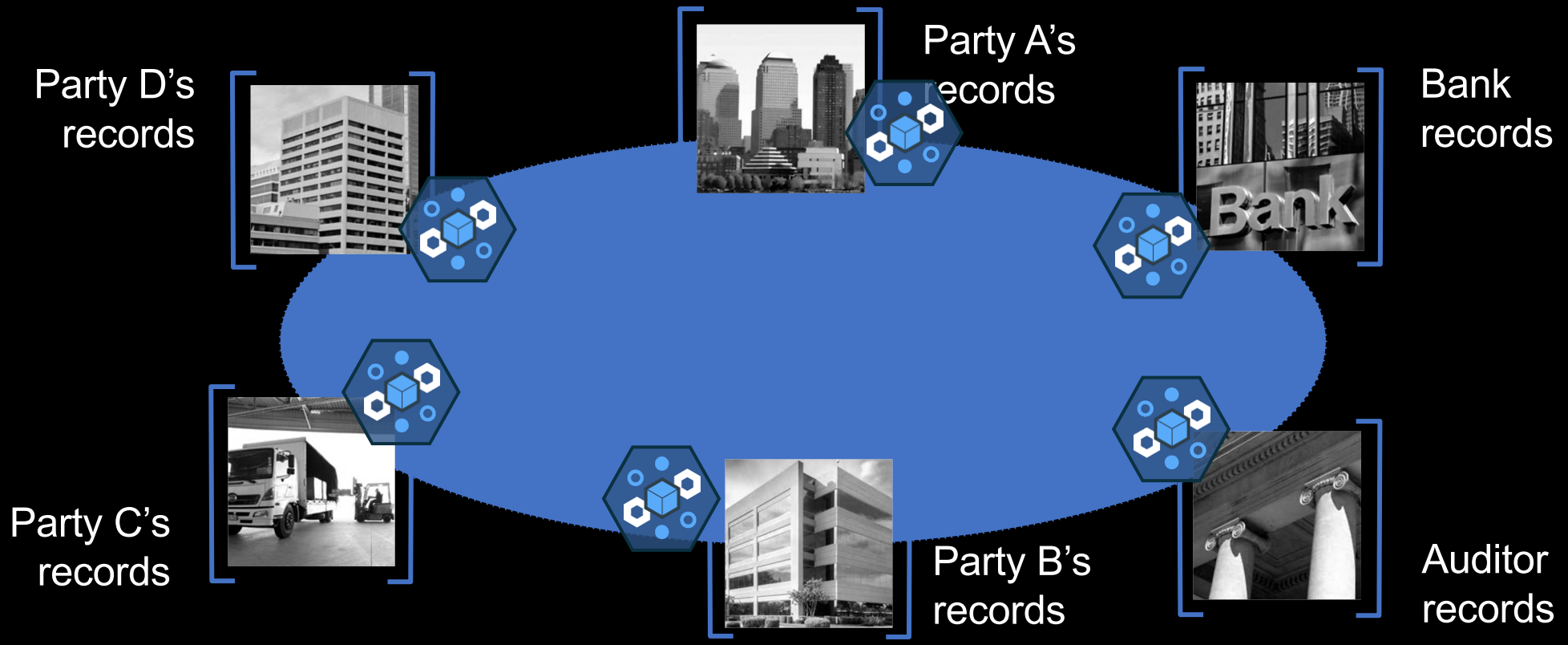
A transaction processing system allowing mutually-distrusting participants in a business network to work with one system of record



Added Value: Distribution of Trust, Transparency, Automation, Governance, etc…

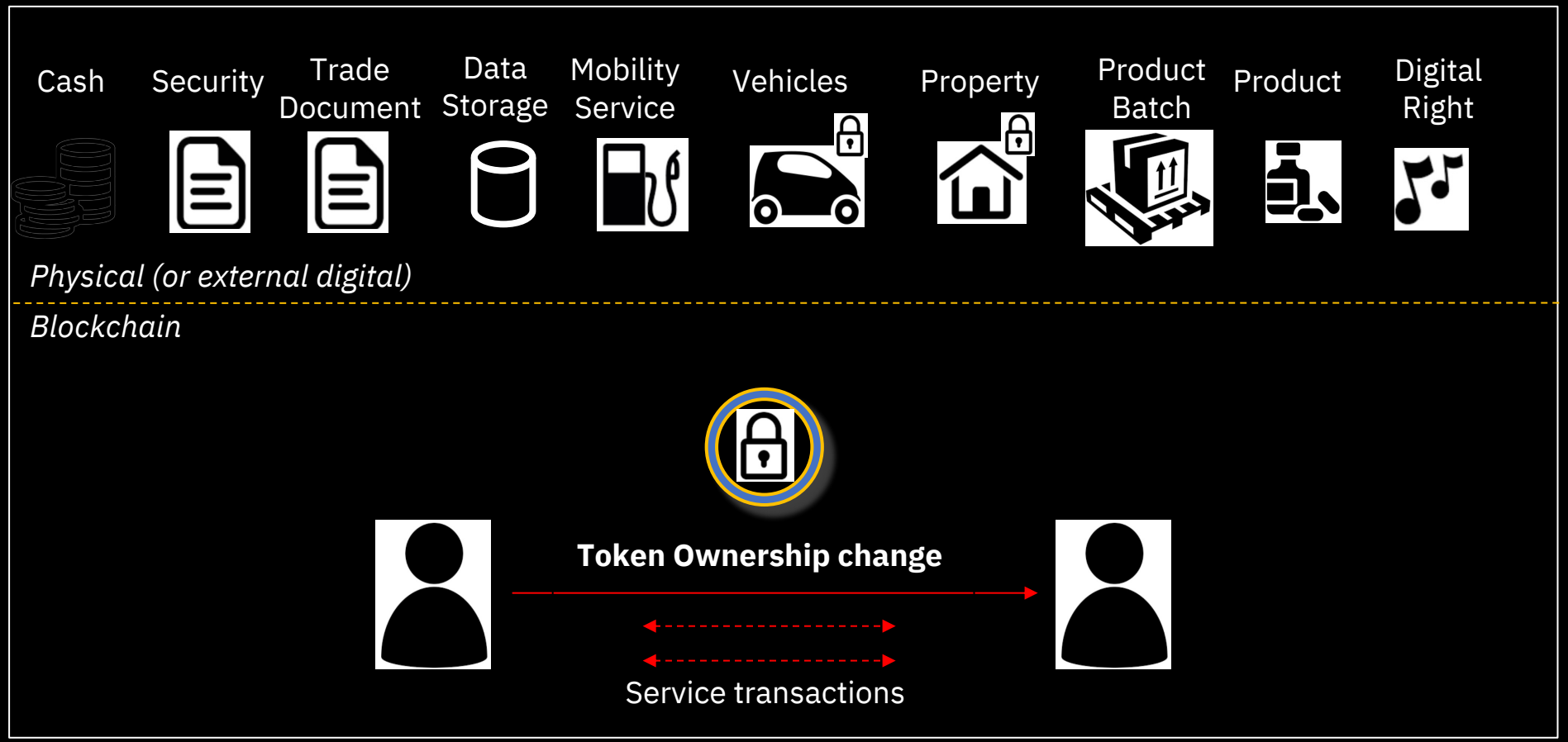
Permissioned Blockchain System: A few observations

A transaction processing system allowing mutually-distrusting participants in a business network to work with one system of record



Combining Transparency w. Privacy & Scalability
Combining Privacy w. Regulatory Compliance & Inhomogeneous Regulation

Management of a Business **Asset** is a Popular Use-Case Pattern

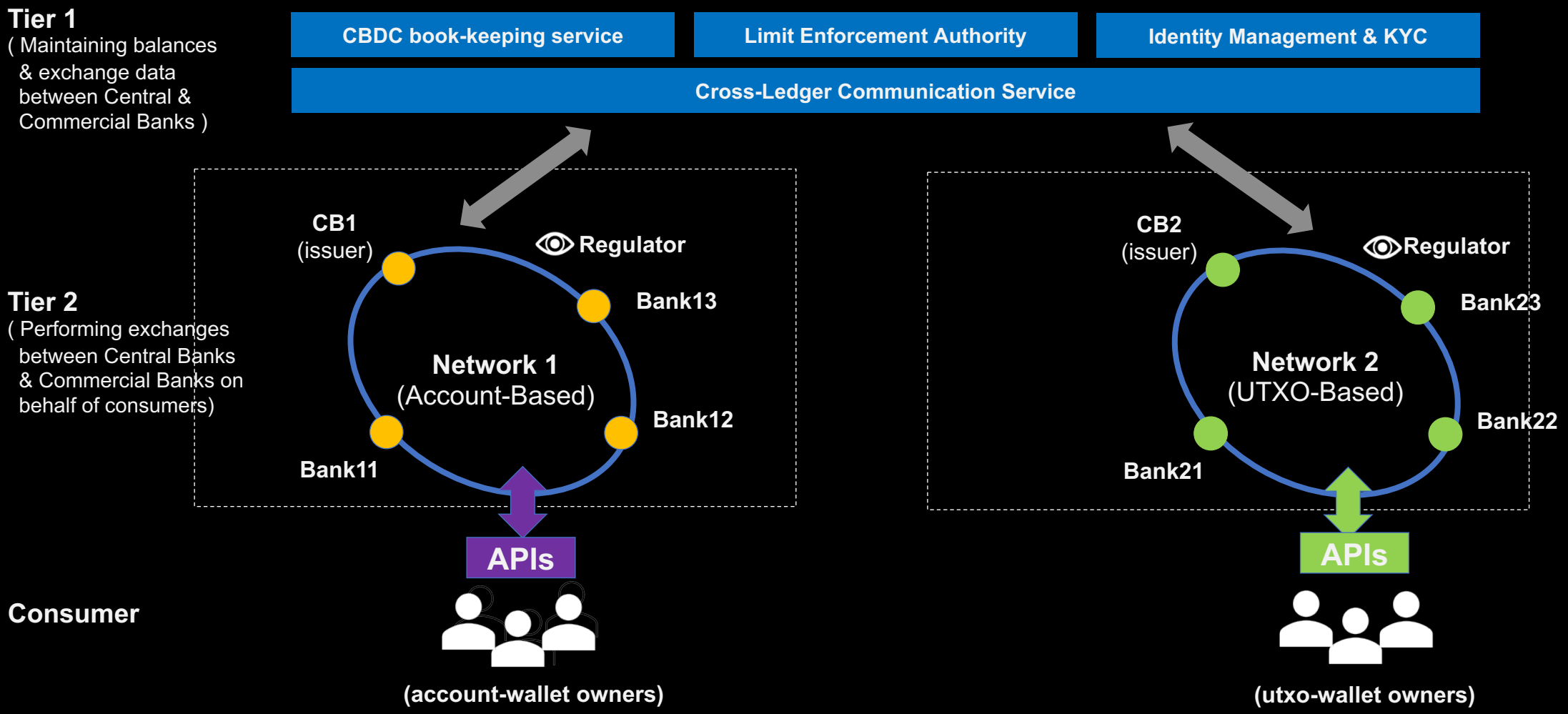


Example use cases

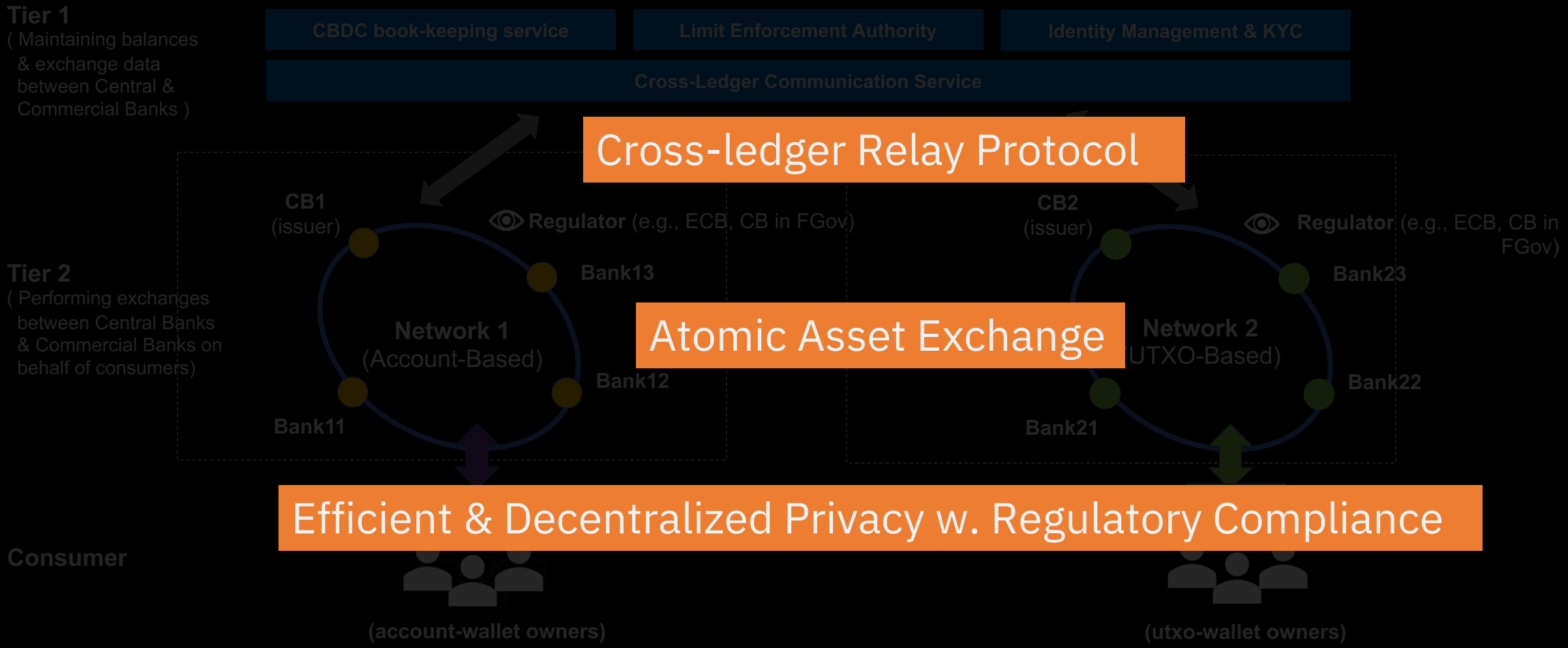
- Securities trading
- Asset transfer
- Digital currency
- Supply chain
- Provenance
- …

- **Token:** a convenient abstraction of a business asset in the blockchain context
- **Tokenization:** the process of representing a business asset as a token in the digital world

Central Bank Digital Currency: A typical architecture



Central Bank Digital Currency: Moving beyond the state of the Art



Experimenting settlement of French government bonds in Central Bank Digital Currency with blockchain technology:

Leading financial institutions publish detailed report on first Central Bank Digital Currency experiment to settle French government bonds

19/10/2021

Paris, 19 October 2021 – A consortium of institutions led by Euroclear have successfully experimented central bank digital currency (CBDC) for settling French treasury bonds on a test blockchain. The experiment, commissioned by the Banque de France included Agence France Trésor, BNP Paribas CIB, Crédit Agricole CIB, HSBC, Societe Generale. IBM provided Euroclear with design expertise and all platform features, including advanced privacy-preserving tokens and hybrid cloud capabilities.

Related report can be found here

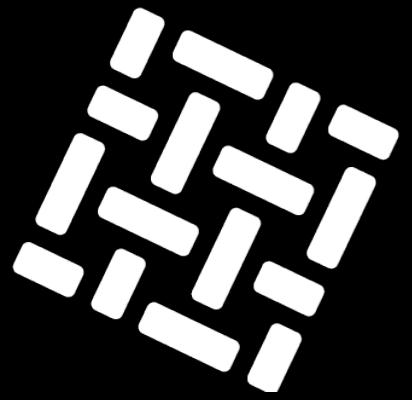
HSBC And IBM Successfully Design And Test Interoperable Multi-Ledger Central Bank Digital Currency

- Cross-border, cross-CBDC and digital currency, cross-asset, cross ledger, end-to-end securities and foreign exchange transactions successfully executed;
- Direct ledger interoperability achieved in a hybrid cloud environment across multiple technologies;
- Demonstration of token-based FX settlement approach may be used for wholesale and retail use-cases.

You can find the related report here.

What's possible today?

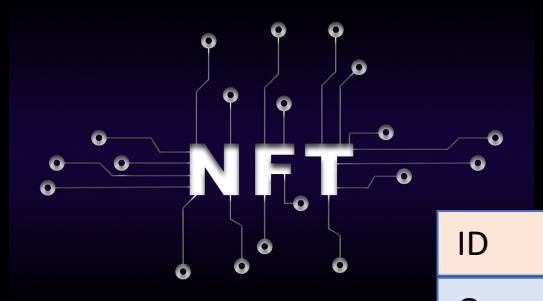
Strengths and Limits



HYPERLEDGER FABRIC

Fabric Tutorials - Non-fungible tokens (NFTs)

Main Fabric tutorials demonstrate how to issue and transfer a NFT that represents an asset



ID	→ Unique identifier for the token/asset
Owner	→ Current owner (organization MSPID or enrolled client identity)
Color Size Appraised Value	→ Token/asset properties

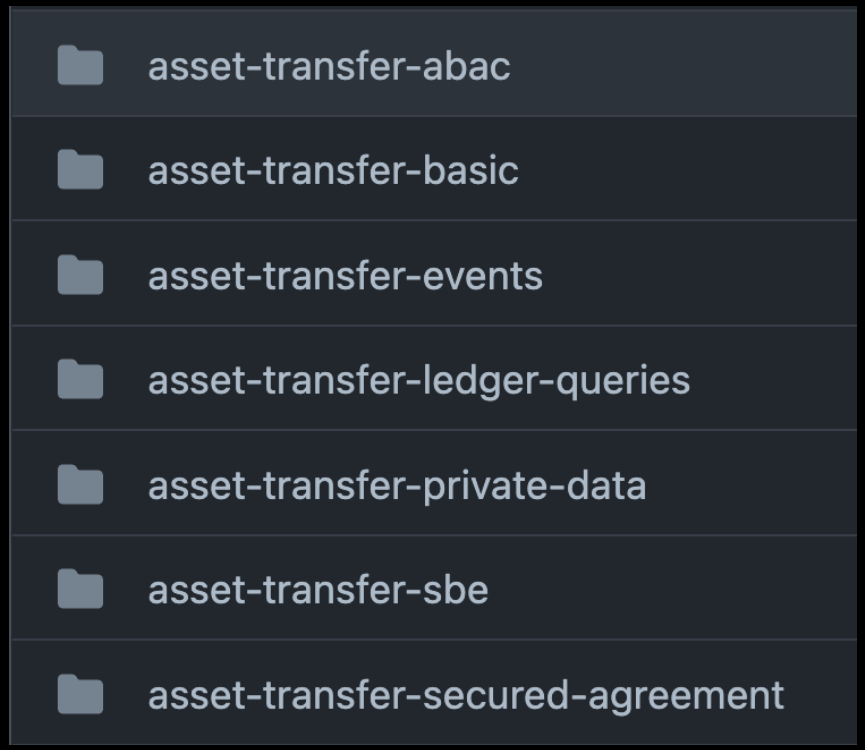
- **Blockchain ledger** maintains token state and owner (in the clear)
- **Smart contracts (chaincodes)** define the rules for token issuance and transfer
- **Endorsement model** defines which organization's peers must execute contract and endorse a token transaction
- **Client applications** for token owner to submit transfers

https://hyperledger-fabric.readthedocs.io/en/latest/tutorials.html

Fabric samples – Non-fungible tokens (NFTs)

Variants of asset transfer NFT sample demonstrate typical token patterns

https://github.com/hyperledger/fabric-samples



Ownership

- Transfer submitter must own token – contract checks enrolled client identity
- Alternatively, submitter's organization must own token – contract checks org MSPID

State-based Endorsement

- Peer from owner org and/or custodian org must execute and endorse transaction

Multi-party agreement

- Clients from all transaction parties must approve a transfer

Privacy

- Use Private Data Collections to keep token data private with data hash on-chain
- Contract checks data against hash before proceeding with transfer
- Endorsing peers disseminate private data to authorized parties
- Alternatively, your application can manage private data and just submit hash to contract

Fabric samples - Fungible and Non-fungible models

Additional Smart Contracts demonstrate various token models

https://github.com/hyperledger/fabric-samples

Fungible tokens

- UTXO (utxo model)
- ERC-20 (account model)

Non-Fungible tokens

- ERC-721

Fungible and Non-Fungible tokens

- ERC-1155

- Ownership - tokens are associated with enrolled client identity
- Transfer - Client can transfer their own tokens
- Check Balance – Client can query peer for owned tokens (no need for separate client wallet)

Fabric Token SDK

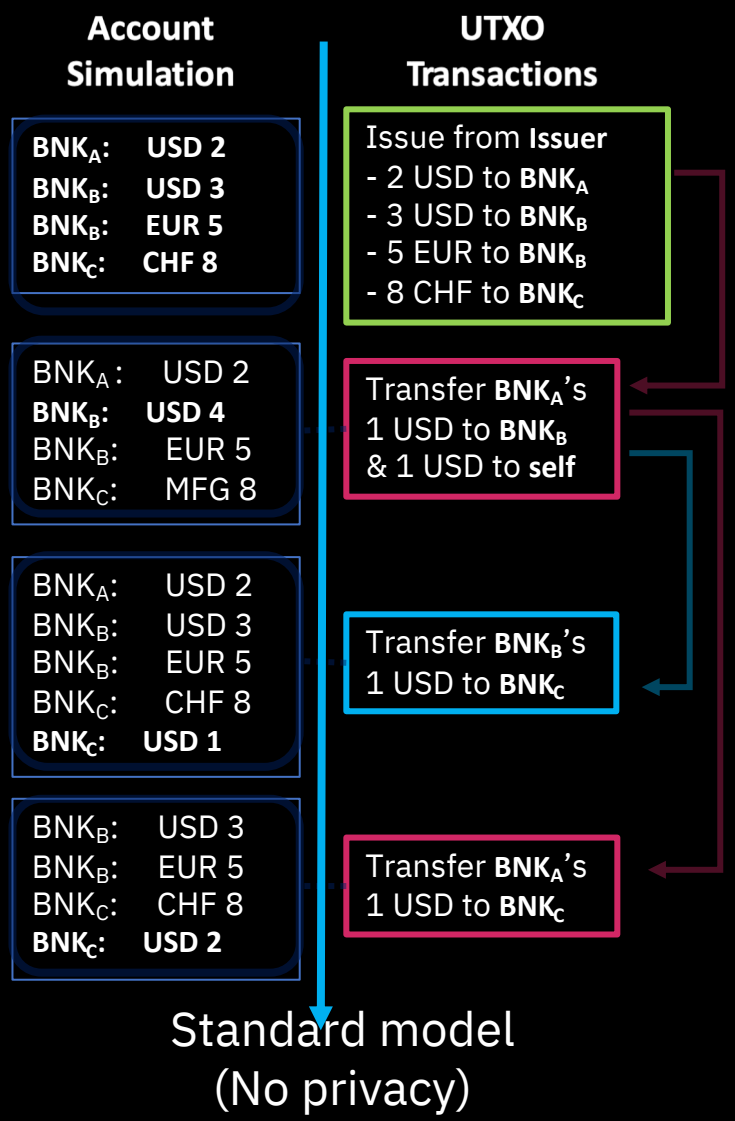
https://github.com/hyperledger-labs/fabric-token-sdk

Privacy and Auditability

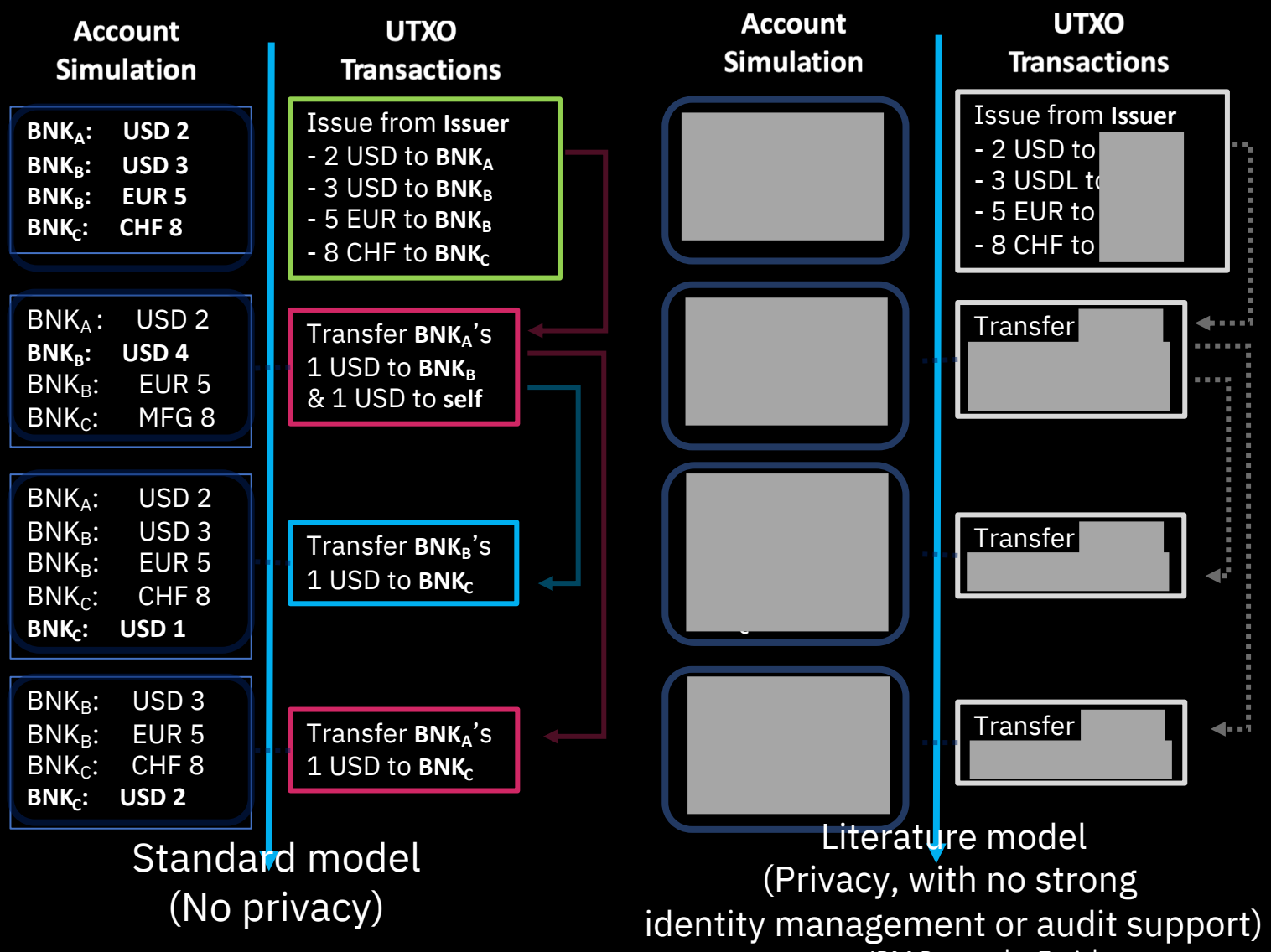
Token SDK Stack

Token Transaction Lifecycle

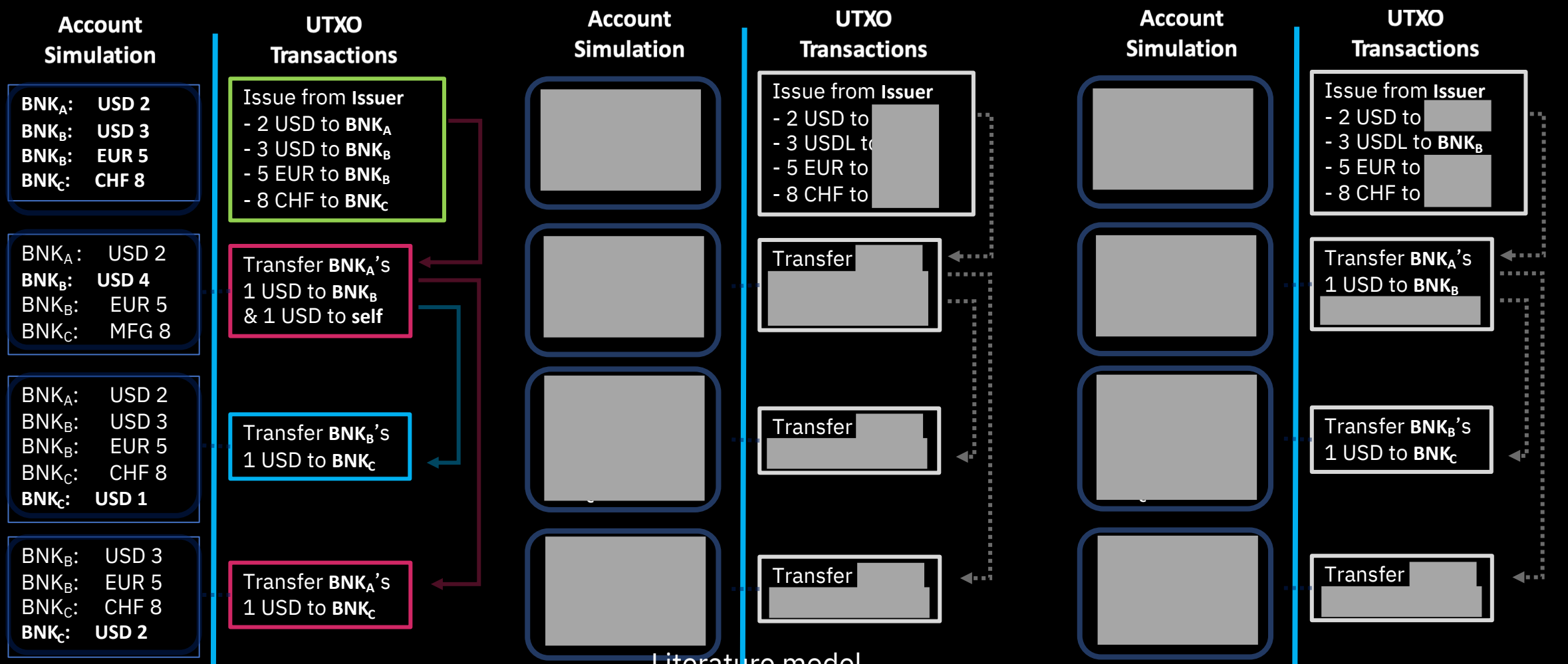
Issuing & settlement w/o privacy



Decentralized Privacy is a Key Priority



Decentralized Enterprise Privacy is a Key Priority



Standard model
(No privacy)

Literature model
(Privacy, with no strong identity management or audit support)

Zero-Knowledge Asset Transfer model
(Privacy w. strong identity management & auditability)

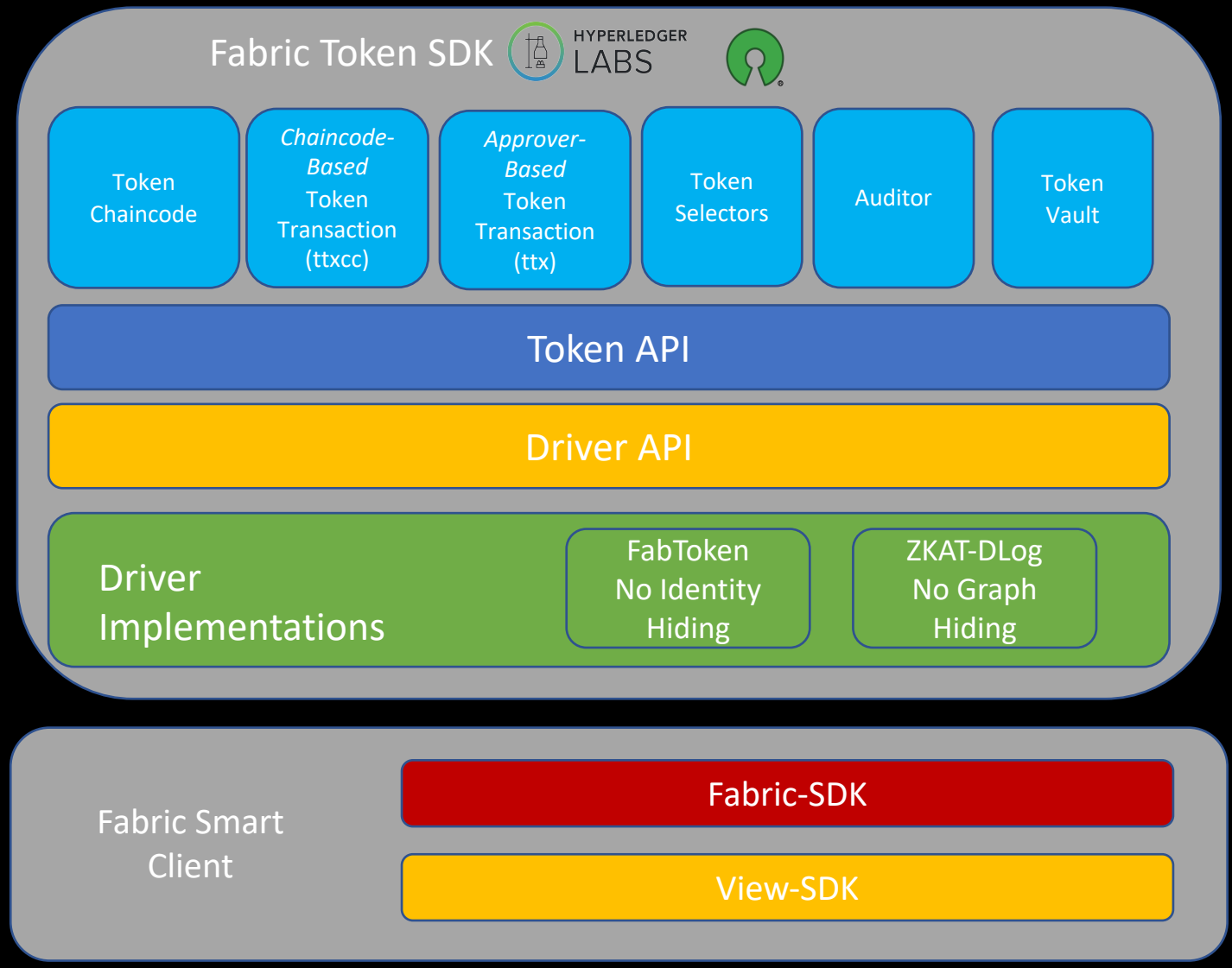
The Fabric Token SDK

- The scope of the Fabric Token SDK is to deliver a set of API and services that let developers create token-based distributed applications on Hyperledger Fabric.
- The Fabric Token SDK has the following characteristics
  - It adopts the **UTXO model;**
  - Key-Management via **Wallets;**
  - It supports **multiple privacy levels**: From an instantiation with no privacy to Zero Knowledge-based instantiations that will obfuscate the content of ledger while enforcing the required invariants;
  - Auditability support;
  - Fungible and Non-Fungible Tokens

A Simple and Effective Token Definition

- A token consists of the following triplet:
  - **Owner**: The owner of the token; Each driver implementation can interpret this field as needed. It can be a public-key, a script, anything the underlying specific driver supports.
  - **Type**: The *denomination* of the token; This is a string whose value can be application specific. Examples are: The denomination of a digital currency or unique identifiers.
  - **Quantity**: The amount stored by this token. It is a positive number encoded as a string containing a number in base 16. The string starts with the prefix 0x.
- These tokens are fungible with the respect to the same type.
- Tokens with the same denomination can be merged and split, if not otherwise forbidden.

A Well-Integrated Family of Stacks

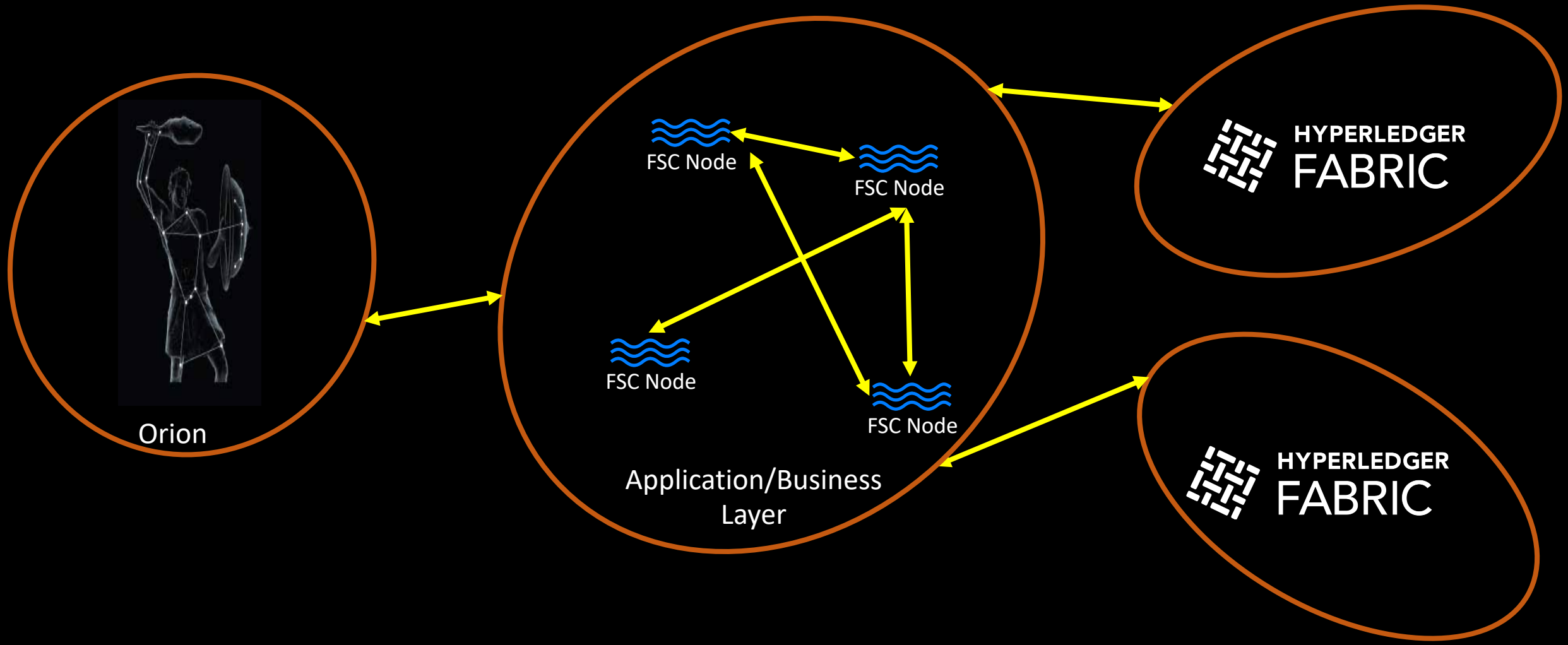


The Fabric Token SDK and the Fabric Smart Client form an Application Layer stack to build token-based distributed applications.

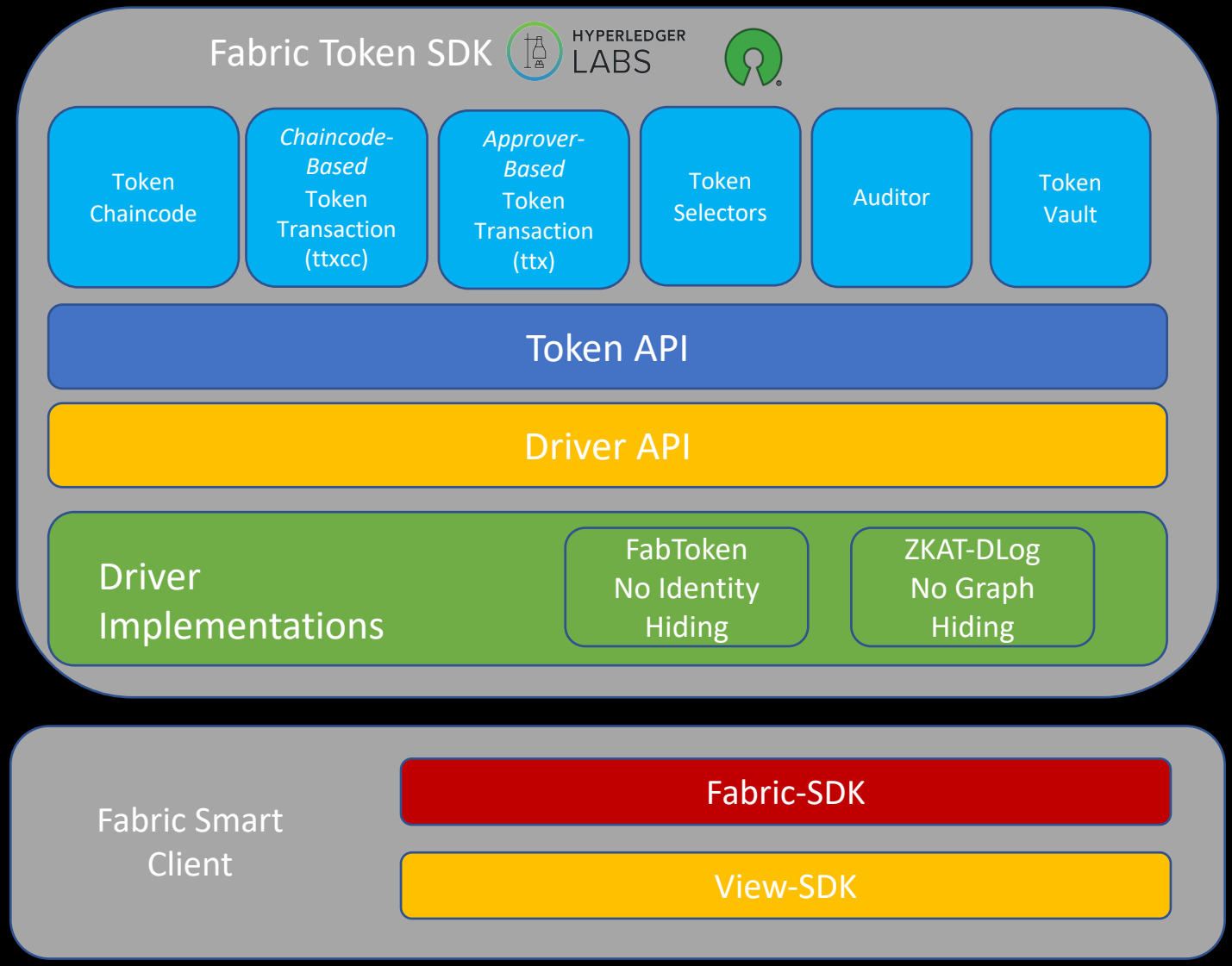
The stack can run as:

- **A standalone network node (Smart Client Node)**

A Network of SFC Nodes as Your Backend Perfect Companion



The Token-SDK Stack is All that You Need

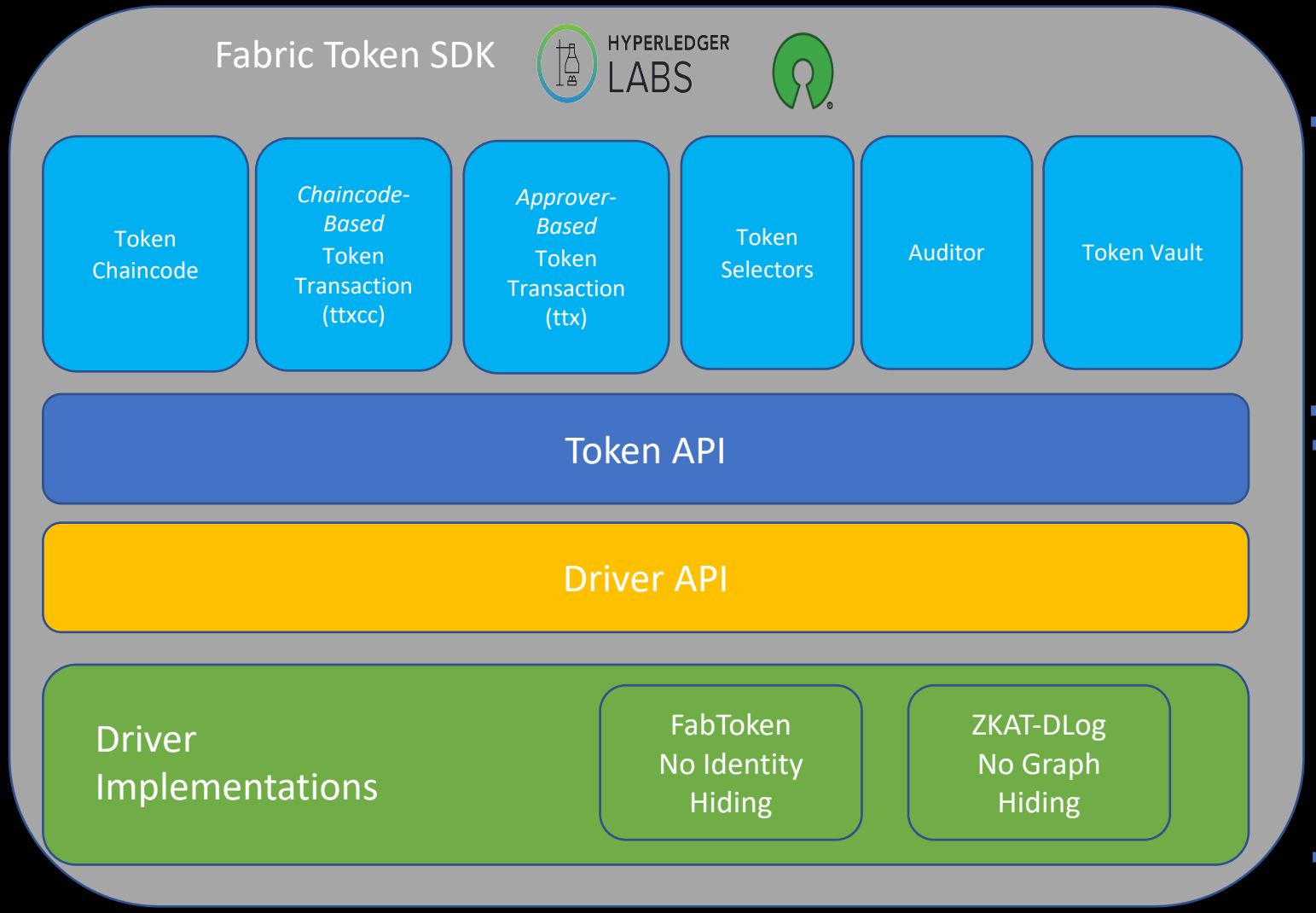


The Fabric Token SDK and the Fabric Smart Client form an Application Layer stack to build token-based distributed applications.

The stack can run as:

- **A standalone network node (Smart Client Node)**
- **Embedded in an already existing Application**

The Token-SDK Stack



- **Services** are libraries built of top of the **Token API** Abstraction to simplify certain tasks (like assembling Token Transactions for Fabric)
- Services are backend specific.
- More services can be provided as third-party add-ons to the Token SDK

- **The Token API** is backend agnostic.
- It uses a meta-representation for tokens that is then translated
- The Driver API defines the contracts any token implementation must satisfy in order to be used by the Token API.
- Multiple drivers with different capabilities.

Anatomy of a Token Request

The **Anchor** is used to bind the Actions to a given Transaction.
In Fabric, the anchor is the Transaction ID.

The **Actions** is a collection of Token Actions.

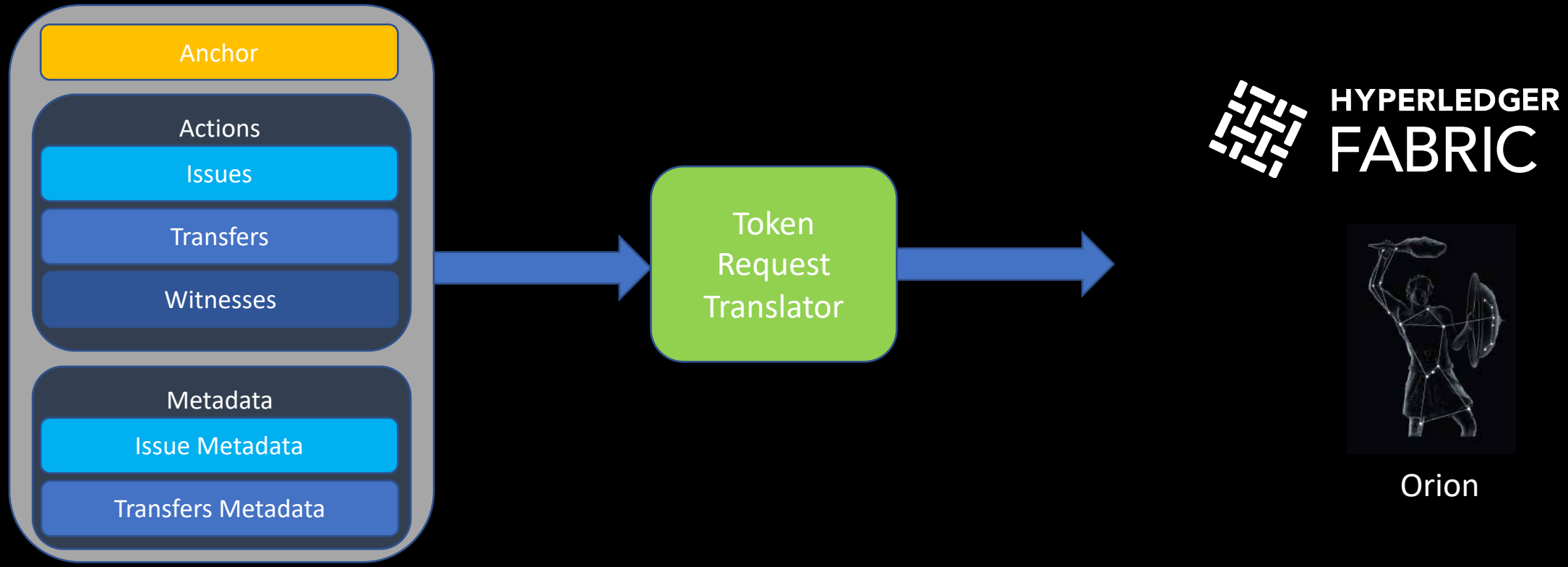
- **Issues** are used to **create new Tokens;**
- **Transfers** are used to manipulate Tokens (e.g. transfer ownership)

The Actions in the collection are independent.

Actions are accompanied by a set of **Witnesses** that attest to the will of the issuers and/or the Token Owners to perform a certain operation.

The **Metadata** is a collection of Token Metadata, one entry for each Token Action. This metadata is exchanged during the Token Request assembling. It contains secret information used by the parties to check the content of Actions collections. The metadata is not stored on the ledger

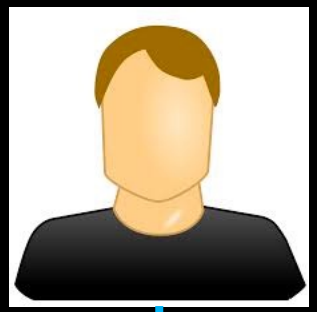
A Token Request is Blockchain Agnostic



A Token Request is translated to the Transaction format of the target backed by a Token Request Translator.

Let's Warmup with an Example: Atomic Swap

Alice to exchange with Bob USD 5 in exchange of CHF 5



Swap CHF 5 with
for USD 5 from Bob

Approve: Swap USD 5
with CHF 5 from Alice

Backend Infrastructure

Atomic Swap Application

Atomic Swap Application

Exchange information

Commit

Token Transaction on Fabric (Chaincode-Based)

(1) A Group of Parties each of which running the Token-SDK in their FSC nodes:

- **Assemble a Token Request**
- **Collect the Required Signatures**
  - One or more Auditors can be involved in the process.
- Then, one of them **submits the Request to the Token Chaincode** to be approved.

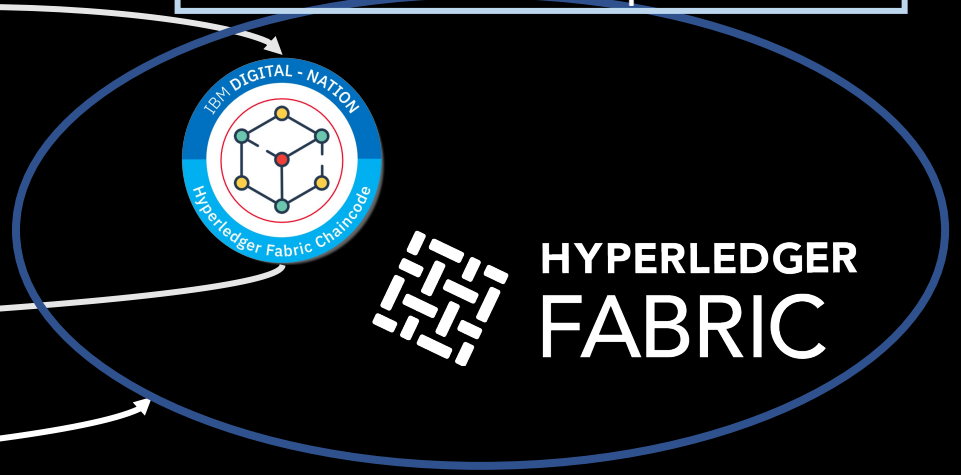
(2) The Token Chaincode, using the Token SDK, does the following:

1. **Validates** the Token Request, and
2. **Translate the Token Request to** RW Set.

Invoke Token Chaincode

Endorsements

Submit Transaction For Ordering



(3) The Fabric Transaction is ready to be sent to the Ordering Service

The Token SDK Makes It Easy to Develop Distributed Application: Just Follow Your Business Process!

```
type TransferFlow struct {
  *Transfer
}

func (t *TransferFlow) Call(context flow.Context) (interface{}, error) {
  // Ask the Token Recipient for the identity she wants to use
  recipient, err := ttxcc.RequestRecipientIdentity(context, t.Recipient)
  assert.NoError(err, "failed to get the recipient identity")

  // Prepare the Transaction
  tx, err := ttxcc.NewAnonymousTransaction(context)
  assert.NoError(err, "failed to create token transaction")

  // Add a Transfer
  err = tx.Transfer(ttxcc.GetWallet(context, t.Wallet), t.Type, []uint64{t.Amount}, []flow.Identity{recipient})
  assert.NoError(err, "failed to add a transfer")

  // Collect signatures
  _, err = context.RunFlow(ttxcc.NewCollectEndorsementsFlow(tx))
  assert.NoError(err, "failed to collect signatures")

  // Send to the ordering service and wait for confirmation
  _, err = context.RunFlow(ttxcc.NewOrderingFlow(tx))
  assert.NoError(err, "failed to order")

  return tx.ID(), nil
}
```

Ask the Token Recipient for the identity she wants to use

Create a new Transaction that will be signed using an Anonymous and Unlinkable Identity (Idemix)

Add a Transfer

Collect all the Required Signatures

Send the Transaction to Fabric and wait its finality.

Token SDK, Fungible Tokens, The Basics

In this Section, we will see examples of how to perform basic token operations like `issue`, `transfer`, `swap`, and so on, on `fungible tokens`.

We will consider the following business parties:

- `Issuer`: The entity that creates/mints/issues the tokens.
- `Alice`, `Bob`, and `Charlie`: Each of these parties is a `fungible token` holder.
- `Auditor`: The entity that is auditing the token transactions.

Each party is running a Smart Fabric Client node with the Token SDK enabled. The parties are connected in a peer-to-peer network that is established and manteined by the nodes.

Let us then describe each token operation with examples:

Issuance

Take Away / Questions

- The Fabric Token SDK delivers a set of API and services that let developers create token-based distributed applications on Hyperledger Fabric.
- It adopts the UTXO model
- Simple Token Definition that can support Fungible and Non-Fungible Tokens
- Simple to use, close to the business logic.

- Find more documentation and samples here:

  https://github.com/hyperledger-labs/fabric-token-sdk
- Reach us on Discord #fabric-token-sdk