Distributed Digital Asset Services Exchange: Design Specification

Author: Ralph Windsor
Revision: 5
Date: 30 April 2018

Overview
This document describes the specification for a distributed digital asset services exchange where participants can store representations of digital assets and service providers can bid to add value to them. The purpose of this model is to allow any kind of asset to be represented digitally and for every change or interaction with it to be recorded in a transaction ledger so the state it was in at any given point in time can be viewed.

In addition to the design specification in this document, the DAM News article: The Digital Asset Transaction Management System – A Time Machine For Digital Assets also contains some detail on a potential implementation model.

Key entities
The following are the major entities:

- A taxonomy composed of assets
- Relationships between assets which are parent/child by default but can be arbitrary.
- Services between assets (represented as transactions) are the interaction of assets and relationships.
- Contracts which define the rules by which services can be invoked and are deemed to have been carried out successfully.
- A distributed transaction ledger (or blockchain) which records all activity.

Design Principles
The following are the main design principles:

Taxonomy-Oriented Structure

- Every entity, attribute, property or action about an asset must be derived from the master asset taxonomy. There must be no hard-coded settings (other than potentially the node containing the setting)

Assets

- The unique asset identifier will be the block in which the instruction to create the asset is first raised. Any operation to create an asset must only contain a single asset.
- Every asset must be related to one parent (i.e. a child) apart from the core or primary asset which is the first one to be created.
- All assets inherit their ancestor's properties.
- Assets can be one or more types.
- Assets have a default parent/child relationship but further relationships can be defined. Relationships between assets must be one of the defined sub-types of the Relationships asset.
- Each asset should have a unique identifier (this should be globally unique, not just unique to the database, but a numeric database ID can be used as well)
- All assets have a manager who can modify the asset (see users).
- Assets can have a management team as well who have the same rights as the manager.

Asset Versions

- The base entity of the system is an asset version
- Each change to an asset is versioned (including the current asset which is a copy of the most recent version)

- Each version should include a transaction reference (see transaction log)

Asset Attributes

- All assets have one or more attributes.
- Asset attributes can be extended
- Asset attributes are assets themselves and inherit from the core asset record (or another asset attribute) themselves.

Owners

- All assets are owned by users, the default is the user who created them.
- Asset ownership can be sub-divided into an arbitrary number of shares. The default is 1,000,000.
- The rules about ownership of assets is determined by an owner relationship and enforced using an owner contract.
- Every asset has a value which is the price of the share multiplied by the number of shares (the market capitalisation). The default value of every asset is zero. Valuations are expressed using a reference to another asset (e.g. a currency).
- Every asset has a register of owners.

Relationships

- The base (and default) relationship is child. All assets except the root are a child of another asset.
- Relationships are assets and inherit from the core asset, called 'relationships'
- Any kind of relationship can be defined, the rules that determine whether the relationship is permitted or not are contracts.

Services

- Services are transactions between assets through their relationships.
- The definition for a service is an asset.

Contracts

- Contracts govern whether a service using the relationship between one or more assets has been carried out satisfactorily.
- Contracts (and their description, attributes and procedural logic) are assets.
- The cost of using services (and the asset to pay for them) is defined in the contract.

Users

- Users are assets and inherit from the core asset called 'users'
- Users have permission attributes

Permissions

- Permissions are assets and inherit from the core asset called 'permissions'

Transactions
Every change to an asset is recorded in the transaction ledger. The transaction ledger is composed of:

- A transaction ID
- A transaction unique identifier
- A pointer to the previous transaction (that transaction's identifier)
- The API call used for the transaction (text)
- The representation of the asset(s) changed as part of the transaction (versions)

The transaction ledger is implemented as a distributed ledger (i.e. a blockchain). Every participating node has a copy of the ledger.

Interface Implementation

- All interfaces should be implemented using API-First principles and the interface should be fully separated from the server-side logic.
- Every action should go through the API.
- Every API action should be stored in the transaction ledger.

Integration With Existing Schema
One possibility of this model is to use the global taxonomy to hold representations of other schema (e.g. semantic web implementations like schema.org). The digital asset exchange should be schema-agnostic and its design is intended to allow it to hold any or all of them. It should be noted that the market (i.e. users and service providers) will determine which actually get implemented and subsequently used (or not) based on supply and demand for them.

Coinless Implementation
To process transactions on the distributed ledger, users will advertise their need by placing an offer to process them. The remuneration for this will be in any asset the user wishes (as long as the service provider agrees to it).

Those who wish to service transaction processing requirements can bid to do so and the user may accept them. The fee is brokered entirely between user and service provider. The digital asset service exchange should not issue its own coins/tokens (although another token could be used as a form of remuneration if both parties agree to it).

The transaction ledger's only purpose is to maintain immutability. There are no maximum nodes defined. A design objective is that the cost of transaction processing should be deflationary (e.g. like storage capacity). The purpose of the exchange is to facilitate commerce, not create alternative means of value transfer.

Trust-less Architecture With A Regulatory Framework
The use of a blockchain avoids the need for a central regulatory body since the network itself enforces compliance. With that said, a regulatory framework (which is optional rather than mandatory) allows best practices to be proposed and users can elect to transact with service providers who adhere to their standards. Further, service providers can potentially increase the level of trust in them by agreeing to audits by a regulator. The objective is to have the benefits of a trust-less blockchain model with some of the benefits of a classic regulated trading exchange.

Hybrid Or On/off-Exchange Implementation
In the initial stages, the majority of interactions with digital assets will not occur via the exchange. As such, it should be possible to carry out transactions either on the exchange or externally to it. There will be issues with missing data by doing this, however, users should not be forced to do everything using the exchange itself. In the same way as cloud services do not require that every interaction is cloud-based, the same should apply here. For this reason, use-cases like DRM are unlikely to be practical because assets will probably not get originated (nor remain) on the exchange.

A further issue is performance, it may take some time to optimise interactions using the distributed ledger which make it impractical for some tasks. The objective should be to minimise and eventually eradicate these limitations over time, but that is an objective, not something that should be mandatory.

Benefits

- A shared global taxonomy vastly simplifies interoperability without needing to enforce a pre-defined schema. If an existing schema is in-demand, it will get used, if not it won't.
- Since all nodes are ultimately interconnected and used by all clients, interoperability can be achieved by referencing the location of the node. The notation for doing this can be defined later, but something like dot or slashes, or a more formal representation using XML could be used.

- A trust-less digital asset services exchange using a contractual model avoids the need for an Enterprise Service Bus. Users of services can exchange data with anyone who agrees to work with them, independently of anyone else (or with their collaboration).
- This kind of structure creates a ready-made digital asset supply chain. Simply using it makes it possible to have complete visibility and transparency about where assets originated from and who did what to them.
- All assets are owned by someone (and there can be many owners who have a share in a given asset).
- The service element is currently under-represented in most blockchain-based digital asset initiatives. Most blockchains exist to support ownership of tokens which are self-serving and are effectively an alternative form of crowd-venture funding. The objective of this model is to allow conventional commercial service providers to interact with each other using whatever form of value exchange they wish (including conventional fiat currencies).

Use Cases
The service exchange can be used for any kind of digital asset (including digital representations of physical assets). Below are a few examples:

- Content Digital Asset Management (images, videos, documents etc)
- New/emerging digital assets (Virtual Reality, Big Data, Internet of Things).
- Digital representations of existing physical assets (e.g. plant machinery etc)
- Financial assets (stocks, bonds etc)
- Alternative assets (artwork, wine, stamps, membership of clubs/associations etc)

Clearly while some of the above are technically feasible, regulatory compliance constraints might make them impractical, however, the model should be theoretically applicable to any kind of asset that can be digitally represented.

Next Steps
Below are some of the issues to be considered next:

- Feasibility of the model?
- Limitations of the model?
- Build from scratch or use an existing technology (e.g. Ethereum etc)
- Achieving adoption
- Test cases and examples.
- Practical implementation issues and how to transition from a partial to a full implementation (e.g. using a combination of on/off blockchain techniques).
- Standards for exchanging asset metadata packets (or lack of them)?