paperchain

Digital Asset Liquidity Protocol

Daniel Dewar, Rahul Rumalla, David Tomaselli

Version 0.3 June 27, 2018

Abstract

Most digital assets are subjected to continuous or limitless consumption upon which economic payouts are calculated by Service Providers. The Digital Asset Liquidity Protocol (DALP) allows for tokenization of these proofs of consumption into Non-Fungible Assets for the purpose of achieving liquidity in a decentralized ecosystem. The protocol aims to build an open standard and a shared interface upon which marketplaces, exchanges, investors and sellers can exchange these tokenized Non-Fungible Assets. With a dual token model, the protocol allows peers in a decentralized network to trade a Non-Fungible Asset-Backed Token (T_{NFT}) that represent the Proofs of Consumption as Non-Fungible Asset(s) in exchange for its notional value in economy tokens (T_{ERC20}). Upon the creation of the Non-Fungible Token, $T_{NFT,}$ it can be either be held until the payment remittance event from the Service Provider or traded in the marketplace in exchange for faster liquidity. What the internet did for information blockchains do for transactions, consequently making an individual transaction a Proof of Information event on the internet. This is the philosophy that drives DALP to capture digital media asset Proofs of Consumption to create a transparent network of media monetization and faster liquidity.

1) Introduction

Traditionally, high-yield assets accrue value over a long period of time for investors, generally meaning that the asset-holders are unable to quickly capitalize on the value of their position. These traditional financial assets also have the challenge of illiquidity—they cannot be easily bought and sold, which creates capital freeze for holders.

This challenge is doubly extended to non-fungible assets that have never historically had investment markets available to them due to the asset nature. Some examples of high-yield accrual based non-fungible assets can be licenses, copyrights, patents, and loans.

2) Proof of Consumption

The massive consumer shift to digital media consumption has transformed the media industry from a manual, physical, product-based business to a digital, transaction-based business with a drastically narrowed bottom-line. However, the media content's transaction data, which can be streams on Spotify, views on YouTube, downloads on the App Store, spins on a Radio Station, or impressions of an Advertisement, are all available in near real-time but go largely unutilized in support of core business processes, in particular liquidity. These transactions are *Proofs of Consumption* that can be tokenized into Non-Fungible Assets in exchange for liquidity in a decentralized eco-system. Proof of Consumption may be categorized into 2 different models, distinguished by how the transactions are converted to payouts.

2.1 Continuous Model

The Continuous Model assumes there is a continuous payout system that is proportional to the quantity of Proof of Consumption transactions. This model can be applied to the payout system of Service Providers like YouTube, Spotify or iTunes App Store where more streams, views or downloads would result in higher monetization.

2.2 Fixed Model

In the Fixed Model, content is under a fixed monetization contract with a Service Provider, which does not take into account the individual consumption transactions. For example, video content licenses on Netflix are subject to a fixed compensation over a fixed period of time, regardless of how many views the content gets. In this model, the Proof of Consumption could be the contract or the license itself.

2.3 Structure

The composition of a Proof of Consumption is generic enough to capture and normalize the transactions coming in from a source provider. The structure of its unit being represented as follows:

- `Source`
- `Period (Start / End)`
- `Asset Type`
- `Metric Type`
- `Units`

- `Price Per Unit`
- `Metadata`

Period	Source	Asset Type	Metric Type	Units	Price	Metadata
July 10th	Spotify	Music	Stream	100	$0.0004	Subscription, Territory, Duration
July 11th	YouTube	Video	View	1000	$0.00007	Subscription, Territory, Duration
July 12th	iTunes Store	Apps	Download	10	$0.70	Territory

Table 2.3.1: An example of Proofs of Consumption in the modern music industry

2.4 Sourcing

Fetching Proof of Consumption transactions from its data sources can occur with the help of an oracle. From *Off-Chain providers,* the process might be straightforward sourcing it in the form of flat files and through API endpoints. However, the *on-chain providers* would need to leverage an on-chain oracle.

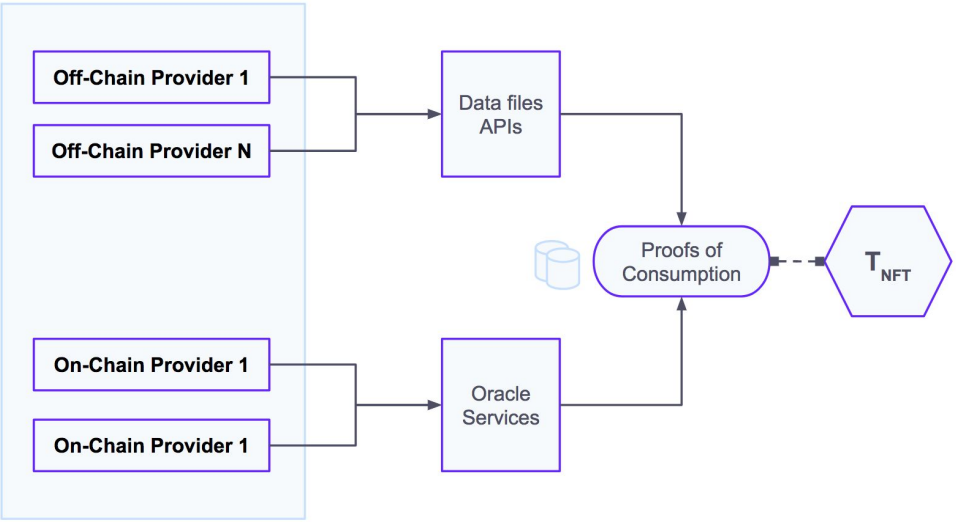

Figure 2.4.1: Visualization of sourcing Proof of Consumption data from off-chain and on-chain providers

3) Digital Asset Liquidity Protocol

The Digital Asset Liquidity Protocol (DALP) enables tokenization of Proofs of Consumption into Non-Fungible Assets in order to achieve liquidity in a decentralized network. The assets are represented by a Non-Fungible Token (NFT), enabling peer-to-peer exchange. Through this protocol, Service Providers are able to tokenize their usage transactions into NFTs while also transparently accounting for the associated monetary payouts to the Content Owners. Content Owners have the ability to hold on to the NFT until the eventual payment remittance or use a Liquidity Marketplace to trade for faster liquidity. The protocol proposes a dual token model.

- **T_{NFT}** - An ERC-721 token to represent the Non-Fungible Asset(s) that are the Proofs of Consumption
- **T_{NFXC}** - An ERC-20 token that is the notional value of the Non-Fungible Asset(s)

While the main purpose of the T_{NFT} is to facilitate the issuance of an IOU and to capture the ownership of the asset, T_{NFXC} is the Non-Fungible Exchange Token i.e., an economy token for purpose of trade.

3.1 Liquidity Marketplace

The marketplace is an implementation of the protocol, in this instance, a decentralized application (dApp) on the Ethereum network.

3.1.1 Actors

In decentralization models where centralized authorities are disintermediated, it is ideal that the responsibilities of the disintermediated role must be assumed by the actors within the networks. In the traditional digital media supply chain, Content Owners are subjected to numerous intermediaries that delay their payment remittance. The 4 primary actors that communicate in the DALP protocol are:

- Service Providers
- Oracle
- Asset Owners
- Buyers

The Proofs of Consumption are packaged and delivered by the *Service Providers*.

An *Oracle* is essentially a dApp responsible for tokenization and liquidation of the assets while also issuing new NFTs.

Asset Owners are identified as the actors that own the content and their tokenized Proofs of Consumption i.e., the Non-Fungible Assets.

A *Buyer* participates in the *Liquidity Marketplace* to assume temporary ownership of the NFT in exchange for faster and perhaps discounted liquidity.

3.1.2 Marketplace Incentive Matrix

The protocol ensures incentives for each of the actors for their participation in the Liquidity Marketplace. To better present this, the following matrix defines the incentive structure:

	Network Value Add	**Incentive**
Asset Owner	Sells tokenized asset as orders $T_{NFT}(O_i)$	Earns T_{NFXC} for sale of T_{NFT}
Buyer	Liquidates orders by purchasing $T_{NFT}(O_i)$	Discounted purchase earning T_{NFXC} returns Resell T_{NFT} in secondary markets
Supplier	Supplies Proofs of Consumption Payment remittance	Earn T_{NFXC} for supplying data
Marketplace ***Oracle***	Tokenizing assets Order settlement	Earn T_{NFXC} in transaction fees

Table 3.2.1: The matrix showing different actors, their roles and their incentives using the Non-Fungible Liquidity Protocol

3.3 Primary Market Liquidity

In the example below, an Asset Owner (Seller) and a Buyer have negotiated and reached agreement to trade an order representing the tokenized Non-Fungible Assets which are the Proofs of Consumptions from a Service Provider. The Exchange Contract is an Ethereum smart contract that executes the functions. The general process is:

1. Buyer and Seller agree on Order terms (For example, discounted liquidity)
2. Buyer’s funds are sent to Seller.

3. Seller receives funds.
4. Buyer assumes ownership of NFT as an IOU.
5. In due time, remittance payment is made by the Service Provider.
6. Funds are redirected to the Buyer.
7. NFT is burned and the Liquidity Transaction is completed.

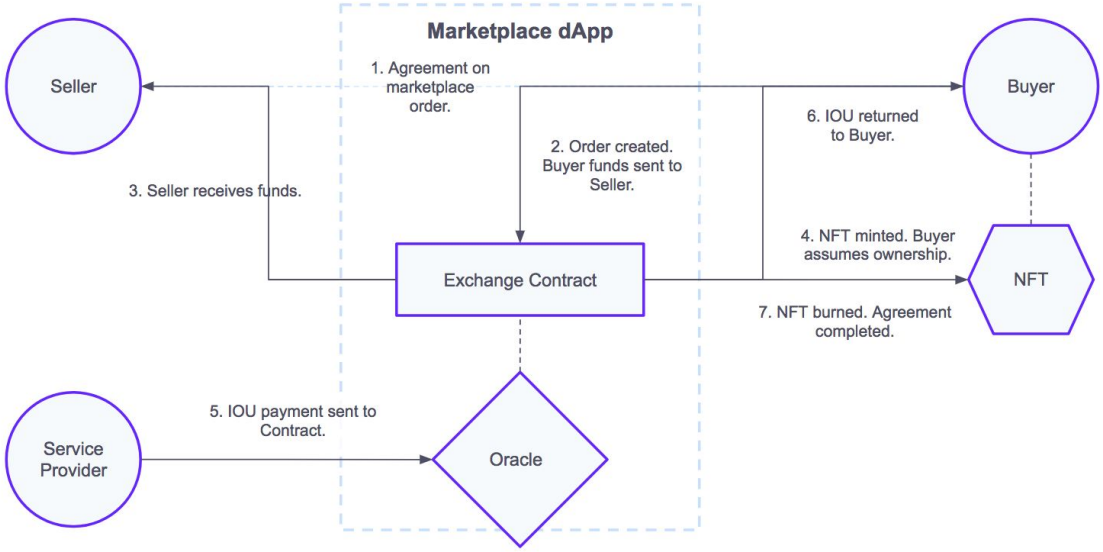

Figure 2.3.1: Example of an order execution sequence between two peers.

2.4 Secondary Markets

In the second scenario shown below, the initial Buyer decides to resell the NFT on a secondary market, causing adjustment in ownership and IOU remittance. In a good Secondary Market scenario, the process is slightly altered:

1. Buyer and Seller agree on Order terms (For example, discounted liquidity)
2. Buyer's funds are sent to Seller.
3. Seller receives funds.

4. Buyer assumes ownership of NFT as an IOU.
5. Buyer and Secondary Buyer reach agreement on re-sale of NFT.
6. Secondary Buyer funds are transferred to Buyer, along with the NFT.
7. In due time, remittance payment is made by the Service Provider.
8. Funds are redirected to the Secondary Buyer.
9. NFT is burned and the Liquidity Transaction is completed.

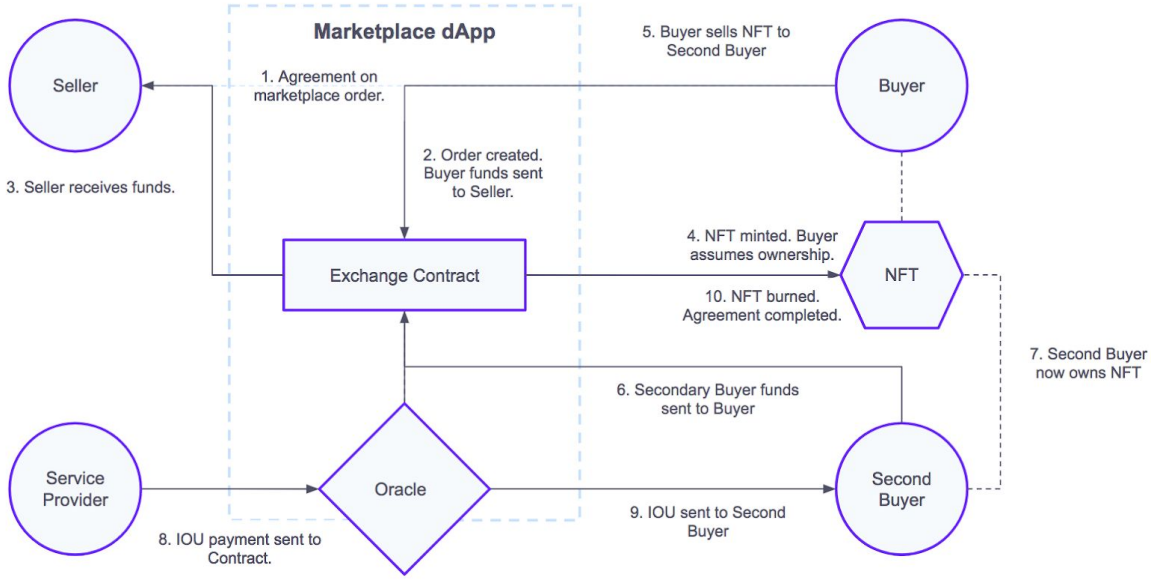

Figure 2: Example of an order execution sequence with a secondary buyer scenario.

4) Technical Design

4.1 Non-Fungible Exchange Contract

The Non-Fungible Exchange Contract (NFXC) is the main smart contract that enables decentralized markets to be deployed on Ethereum. It is an extension of both the ERC-20 and the ERC-721 token contracts, where the ERC-20 token T_{NFXC} is the

economy used to trade and the ERC-721 token T_{NFT} represents the temporary ownership of a collateralized Asset.

4.2 The Mechanics of the Contract

`orders[...]`

An order is the atomic unit of trade in the market(s) created by NFXC. The Seller sells an order whereas the Buyer buys an order to assume ownership of receivable. An order has the following characteristics:

- `orderId`
  The unique index of the order
- `orderAmount`
  The value of the order represented in T_{NFXC} (ERC-223 tokens)
- `chargeAmount`
  The fraction of the total amount which the Market charges as a charge fee
- `returnAmount`
  The fraction of the total amount which the Buyer makes as a return
- `seller`
  The wallet address of the Seller
- `buyer`
  The wallet address of the Buyer
- `fundingDeadline`
  The timestamp indicating by which the seller desires the order to be funded by
- `remittanceDue`
  The timestamp, set by the Oracle, indicating when the expected remittance is due

- `status`

  The order status indicates the stage of the order lifecycle
  - *Published* when the order is published by the seller
  - *Funded* when the order is funded by the buyer
  - *Remitted* when the order amount is remitted by the Oracle
  - *Expired* when the order was not funded by the desired deadline
  - *Cancelled* when the published but non-funded order was cancelled by the seller
- `oracleCorrelationId`

  Set by the Oracle to uniquely identify the order. The buyer would be able to verify the uniqueness using the oracle's off-chain systems using the pre-configured API endpoint.

`deployMarket(name, address, chargePart, oracleEndpoint)`

When a market is deployed using the NFXC, it needs to be supplied with the parameters necessary for the orders and trades to take place. The deployer of the market assumes ownership of the market. This is done by the marketplace Oracle.

- `name`

  The name of the Oracle
- `address`

  The wallet address of Oracle for which all the network fees are sent to
- `chargePart`

  The fraction (of 1000) of the total amount configured as the network charge fee. For example 0.5% is represented as 5 (i.e., 0.5% of 1000) and the Oracle makes 5 T_{NFXC} for an order that's worth 1000 T_{NFXC}

- `oracleEndpoint`

  The public API endpoint URL for which orders on-chain can be related to the off-chain metadata and Proofs of Consumption packages

```
publishOrder(seller, amount, deadline, remittanceDue)
```

Alice is the **seller** that publishes an order with token **amount** *10,000* T_{NFXC} on *July 1st 2018*, for which she is hoping to receive funding by the **deadline** July 28th 2018. She indicates that she expects the buyer, Bob, would receive his funds by the expected **remittance due date** September 1st 2018.

When Alice published the order of 10,000 T_{NFXC}, the NFXC splits Alice's order into 3 parts using Oracle configuration variables:

- Charge amount of 50 T_{NFXC} (0.5% of 10,000 set by Oracle during deployment)
- Return amount of 250 T_{NFXC} (2.5% of 10,000 set by Oracle during deployment)
- Order amount of 9,700 T_{NFXC} (Original amount - (Charge + Return))

Should the order be funded, *Alice* receives 97% of the original posted amount.

```
fundOrder(buyer, orderId)
```

The **buyer**, Bob, sees Alice's order indicated with its orderId on the marketplace and likes the fact that he can make 2.5% return by the expected remittance due date and he is happy to take that risk for a short term return.

When Bob funds the order, Bob's wallet has now been debited with 9750 ERC-223 token T_{NFXC} and credited with 1 newly minted ERC-721 token T_{NFT}. The T_{NFT} is the

ERC-721 token that is tied to the order that is indicative of Bob's ownership of the order. The $T_{NFT}(O_i)$ is the IOU from Alice to Bob.

From the 9750 T_{NFXC}, 50 T_{NFXC} is transferred to Oracle's pre-configured address as 0.5% charge fee and the remaining 9700 T_{NFXC} is transferred to Alice the seller.

```
remitOrder(orderId)
```

Oracle and/or Service Provider will be authorized off-chain, by the seller Alice, to remit funds to Bob when her funds become available by the previously mentioned *remittance due date.*

10,000 T_{NFXC} are transferred from the Service Provider to Bob's wallet. Bob, now having received the funds, has made a return of 250 T_{NFXC} i.e., 2.5% of the 10,000 T_{NFXC}. The associated NFT with the order i.e., $T_{NFT}(O_i)$ is then burned. This relieves Bob of the ownership and the IOU is considered as settled.

Prior to remittance, Bob is free to re-sell his $T_{NFT}(O_i)$ and transfer his ownership to another buyer. If this is the case, the Oracle will remit the payment to the current owner of the $T_{NFT}(O_i)$.

	Order state	*Alice (0xa1b2c30)*	*Bob (0xq1w2e3)*	*Oracle (0xz5x6c7)*
Publish Order	`orderId` 1 `seller` 0xa1b2c30 `orderAmt` 9700 `chargeAmt` 50 `returnAmt` 250 `orderDate` July 1st '18 `deadline` July 28th '18 `remitDue` Sept 1st '18 `status` Published	0 T_{NFXC}	10,000 T_{NFXC}	*X* T_{NFXC} Where *X* is the token supply in Oracle' reserve
Fund Order	`status` Funded `buyer` 0xq1w2e3	+ 9700 T_{NFXC}	- 9700 T_{NFXC} + 1 $T_{NFT}(O_1)$	+ 50 T_{NFXC}
Remit Order Payment	`status` Remitted		+ 10,000 T_{NFXC} - 1 $T_{NFT}(O_1)$	- 10,000 T_{NFXC}

Table 4.2.a: Account state changes of an order's lifecycle on the marketplace

4.3 Oracle API Specification

One of the main functions of the Oracle is to bridge the gap between the off-chain dApp(s) and the on-chain contracts. Every order on the NFXC is tagged with a `orderCorrelationId` field, which can be used to fetch additional metadata from the Oracle's publicly accessible API endpoint named `getOrderMetadata(orderId)`

```
{
  "orderId": 123,
  "proofOfConsumptionIds": [
    1,
    2,
    3
  ],
  "createdAt": "2018-01-02 16:17:18",
  "createdBy": "0x356787383b3d25c92990c656696e40300e47316b"
}
```

Image 4.3.a: An example of a JSON output from the oracle API.

5) Token Utility

The exchange token T_{NFXC} is a multi-purpose ERC-20 token that allows peers on DALP to transact and to use dApps or any integration services connected to it. Some of the additional uses of the token are:

- All economic incentives in the network are paid out in T_{NFXC}
- Price locking features for Orders on the Liquidity Marketplace. (a % is dedicated to a stability fund that will be used to offset the order at the time of liquidity event(s) to protect the seller from volatility if needed). Here, the idea of 'stability' is a Token Engineering characteristic more than an entire protocol.
- Similar to a credit score, a reputation score is tied to wallet addresses with token transaction history. For example, a good reputation score can lower risk fees.
- For any programmatic trading on the network, a staked pool can help gain priority for competing orders.

A custom token in general allows for decoupling from the parent community (like Ethereum), opening up many more possibilities like customized contributions, reward mechanism and implementing governance structures where or when necessary.

6) NFXC Configuration and Extensibility

While the contract mechanism outlined so far is in a simplistic form, the standard itself allows for some exciting extensibility per the desired marketplace configuration:

- Offering an improved level of price discovery by allowing the buyer(s) to bid for a price ensuring the seller to get the best price on the market. This would imply that the seller withdraws the funds from the buyer on accepting the bid
- For larger valued orders, facilitation of funding via buyer pools where buyers' return staked to their contribution allowing them to spread their risk
- Creating market conditions and incentivization to open up asset price discovery to multiple service registries
- Configuring the market or a deal to incur a maturity interest rate, that is tied directly to the NFT, in scenarios where remittance payback is delayed beyond the expected date. This can be applied to models like advances and interest bearing loans.
- This standard also allows for models implementing a speculative aspect on the futures of seller's assets, thereby creating higher volatility and healthier liquidity on the platform - a zero sum speculative market allowing for higher returns for investors applying their industry and market expertise
- Prediction markets can be tied to the performance of the asset over time
- Incentivising Oracles to connect zero knowledge proof protocols to ascertain transaction data quality, as well as initiate price discovery modelling for data flows

7) Summary

By adopting the Digital Asset Liquidity Protocol, we aim to provide a decentralized mechanism for peer-to-peer trading of tokenized non-fungible assets in primary and secondary markets. We can extend the core framework to include marketplace services to manage off-chain order books, address front-running issues and adopt prediction market mechanism and zero knowledge proof protocols to incentivise wider network adoption.

This version of the Protocol White Paper is intended for technical review, subject to iterative improvements and not for public dissemination. For comments, questions and feedback please contact us at hello@paperchain.io.