paperchain

Asset Liquidity Protocol

Daniel Dewar, Rahul Rumalla, David Tomaselli

Version 0.2 June 22, 2018

Abstract

The Asset Liquidity Protocol enables tokenization of illiquid Non-Fungible Assets for the purpose of achieving short-term liquidity in a decentralized marketplace. The protocol aims to build an open standard and a shared interface upon which marketplaces, exchanges, investors and sellers can trade tokenized and collateralized Non-Fungible Assets. The protocol proposes a dual token model allowing two peers (a Buyer and a Seller), to transact with each other in a marketplace, where the Buyer is issued an IOU, a Non-Fungible Asset-Backed Token (T_{NFT}) that collateralizes the Seller's illiquid Non-Fungible Asset(s) in exchange for a discounted notional value of the Asset in liquid ERC-20 tokens (T_{ERC20}). Upon repayment, the T_{NFT} is burned. All trades get executed on the Ethereum Smart Contracts that are integrated with the Marketplace dApps.

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. In the case of Paperchain, it is digital media content.

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 App Store, spins on a Radio Station, 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 collateralized into NFTs in exchange for liquidity.

This protocol is influenced from financing models of traditional supply chain like invoice factoring and lending.

2) Non-Fungible Liquidity Protocol

Non-Fungible Liquidity Protocol (NFLP) enables peer-to-peer exchange of collateralized and tokenized Non-Fungible Assets for liquidity while also creating secondary market opportunities. This protocol is built upon a dual token model.

- **T_{NFT}** - An ERC-721 token to represent the Non-Fungible Asset(s)
- **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} acts as an economy token for trading and lends itself to governance mechanisms (see section 2.4).

2.1 Actors

As it goes with any decentralization models, when centralized authorities are disintermediated, the responsibilities of the disintermediated role must be assumed by the actors within the networks. The 4 primary actors that communicate with the protocol are defined below:

- Sellers
- Buyers
- Marketplace Oracle
- Suppliers

Sellers are identified as the actors that own the Non-Fungible Assets, which are typically supplied by the *Suppliers*. A *Marketplace Oracle* actor is a dApp responsible for tokenizing the assets and the issuance of the NFT to the actor buying the asset (the *Buyer*).

2.2 Incentive Matrix

The protocol ensures incentives for each of the actors for their roles on the network. To better present this, the matrix defines the incentive structure:.

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

Table 2.2.1: The matrix showing different actors, their roles and their incentives using the Non-Fungible Liquidity Protocol

2.3 Primary Markets

In the example below, a Seller and Buyer have negotiated and reached agreement to trade a non-fungible asset. Agreement between the Buyer and Seller initiates the NFLP. The general process is:

1. Buyer and Seller agree on Order terms.
2. Buyer's funds are sent to Seller.
3. Seller receives funds.
4. NFT is minted and Buyer receives an IOU.
5. In due time, remittance payment is made and recorded by the NFLP.
6. Funds are sent to the Buyer.
7. NFT is burned and the Agreement is completed.

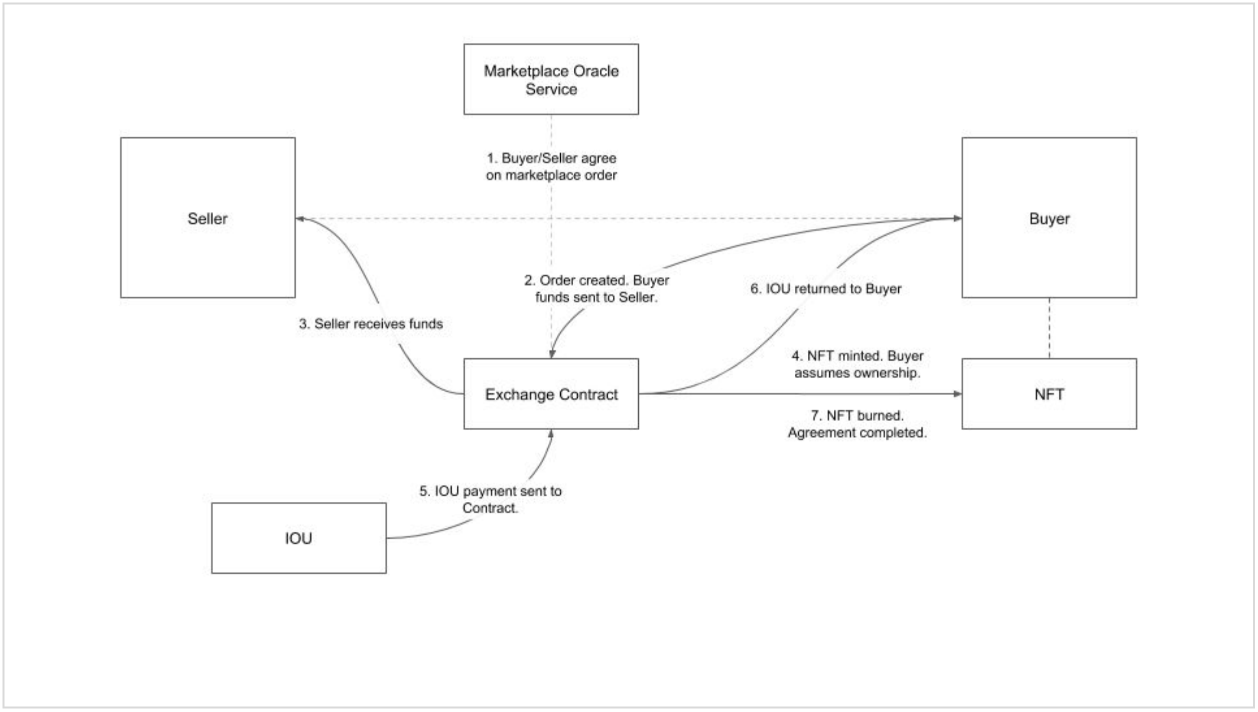
Marketplace Oracle Service
Seller
Buyer
1. Buyer/Seller agree on marketplace order
2. Order created. Buyer funds sent to Seller.
3. Seller receives funds
6. IOU returned to Buyer
4. NFT minted. Buyer assumes ownership.
Exchange Contract
NFT
7. NFT burned. Agreement completed.
5. IOU payment sent to Contract.
IOU

Figure 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.
2. Buyer’s funds are sent to Seller.
3. Seller receives funds.
4. NFT is minted and Buyer receives an IOU
5. Buyer and Secondary Buyer reach agreement on sale of NFT.
6. Secondary Buyer funds are sent to Buyer.
7. The NFT, representing the IOU from Seller, is transferred to Secondary Buyer.
8. In due time, the remittance payment is made and recorded by the Smart Contract.
9. Funds are sent to the Secondary Buyer.
10. NFT is burned and the Agreement is completed.

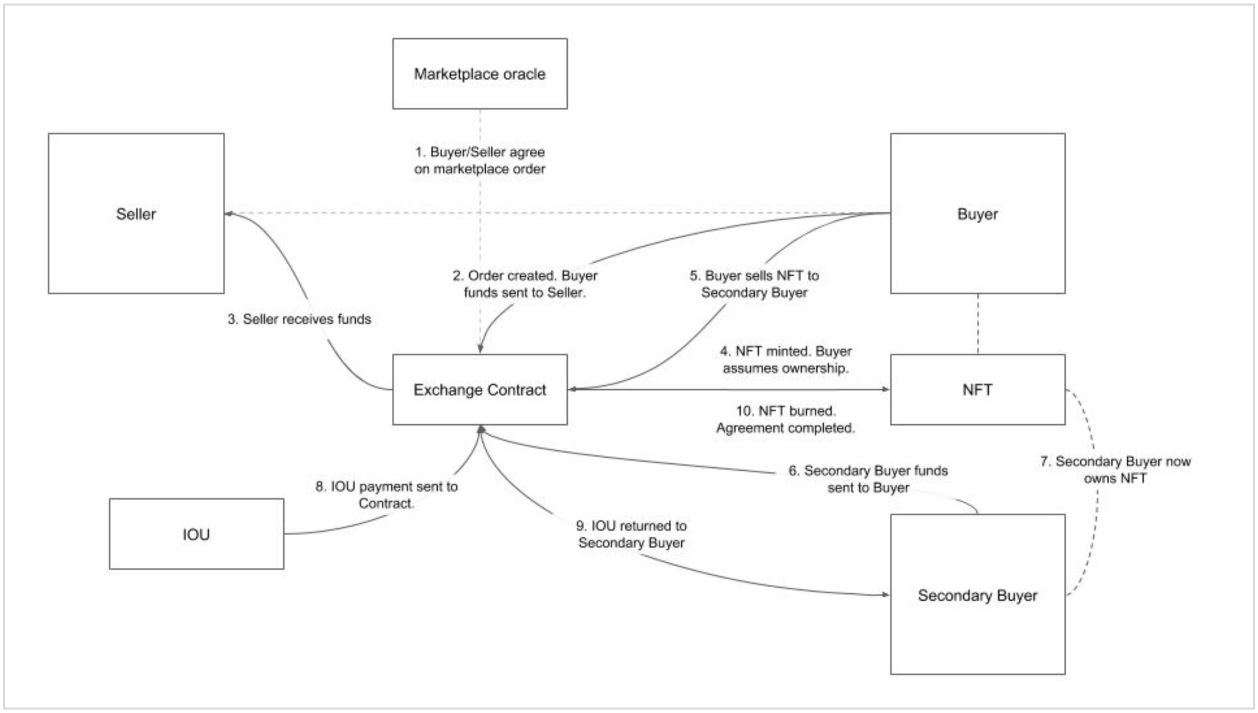
Marketplace oracle
Seller
Buyer
1. Buyer/Seller agree on marketplace order
2. Order created. Buyer funds sent to Seller.
3. Seller receives funds
5. Buyer sells NFT to Secondary Buyer
4. NFT minted. Buyer assumes ownership.
Exchange Contract
NFT
10. NFT burned. Agreement completed.
6. Secondary Buyer funds sent to Buyer
7. Secondary Buyer now owns NFT
8. IOU payment sent to Contract.
IOU
9. IOU returned to Secondary Buyer
Secondary Buyer

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

3) Non-Fungible Exchange Contract (NFXC)

The NFXC is the main contract that enables decentralized markets to be deployed on Ethereum. It is an extension of both the ERC-223 and the ERC-721 token contracts, where the ERC-223 token T_{NFXC} is the economy used to trade and the ERC-721 token T_{NFT} represents the temporary ownership of a collateralized Asset.

3.1 The Mechanics of the Contract

`orders[...]`

An order is the atomic unit of trade in the market(s) created by NFXC. The sellers sells an order whereas the buyers buy 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 MOS, 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 MOS
    - *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 MOS 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, returnPart, mosEndpoint)`

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 MOS entity

- `name`

  The name of the MOS
- `address`

  The wallet address of MOS for which all the network charge 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 MOS makes 5 T_{NFXC} for an order that's worth 1000 T_{NFXC}
- `returnPart`

  The fraction (of 1000) of the total amount configured as the buyer's return fee. For example 2.5% is represented as 25 (i.e., 2.5% of 1000) and the MOS makes 25 T_{NFXC} for an order that's worth 1000 T_{NFXC}

- `mosEndpoint`
  Indicate the API endpoint URL for which orders on-chain can be related to the off-chain entities using the MOS

`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 MOS configuration variables:

- Charge amount of 50 T_{NFXC} (0.5% of 10,000 set by MOS during deployment)
- Return amount of 250 T_{NFXC} (2.5% of 10,000 set by MOS 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 September 1st - 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 MOS's pre-configured address as 0.5% charge fee and the remaining 9700 T_{NFXC} is transferred to Alice the seller.

```
remitOrder(orderId)
```

MOS is authorized off-chain, by the seller Alice to remit funds from Alice to Bob when her funds are available by the previously mentioned *remittance due date.*

As the funds becomes available in Alice's account, the MOS transfers 10,000 T_{NFXC} from Alice's wallet 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 MOS then uses the NFXC to burn the associated NFT with the order i.e., $T_{NFT}(O_i)$. 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 MOS will remit the payment to the current owner of the $T_{NFT}(O_i)$.

	Order state	***Alice (0xa1b2c30)***	***Bob (0xq1w2e3)***	***MOS (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 MOS’ reserve
Fund Order	`status` Funded `buyer` 0xq1w2e3	+ 9700 T_{NFXC}	- 9700 T_{NFXC} + 1 $T_{NFT}(O_1)$	+ 50 T_{NFXC}
Off-chain settlement	-	- 10,000 T_{NFXC}		+ 10,000 T_{NFXC}
Remit Order Payment	`status` Remitted		+ 10,000 T_{NFXC} - 1 $T_{NFT}(O_1)$	- 10,000 T_{NFXC}

4) Marketplace Oracle Service (MOS)

The Marketplace Oracle Service is a dApp that interfaces with the NFXC with the ability to reference the Marketplace Orderbook for matching Contract data to Marketplace data, using a correlation identifier.

Additionally, the MOS functions can be listed as:

- Tokenizing Non-Fungible Assets
- Maintain an off-chain correlation of the Orderbook
- Order settlement with a buyer network
- Remittance facilitation from suppliers to buyers

5) 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 market 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 the funding via buyer pools where buyers' return is calculated by their stake and also allowing the buyers 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 after 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 MOS to connect zero knowledge proof protocols to ascertain transaction data quality, as well as initiate price discovery modelling for data flows

6) Summary

By adopting the NFXC protocol, we aim to provide a decentralized mechanism for peer-to-peer trading of 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.

For comments, questions and feedback please contact us at hello@paperchain.io.