paperchain

Digital Asset Liquidity Protocol

Rahul Rumalla, David Tomaselli & Daniel Dewar

Version 0.4 August 14th, 2018

Abstract

Most digital assets are subjected to continuous or limitless consumption, upon which relative economic incentives are distributed to the asset owners. The transaction data representing the asset consumption is the core proof of the eventual payout. The **Digital Asset Liquidity Protocol** (DALP) allows for tokenization of these proofs of consumption into Non-Fungible Assets for the purpose of achieving liquidity on a decentralized exchange (DEX). The protocol aims to build an open standard and a shared interface upon which marketplaces, exchanges, investors and makers can exchange these tokenized Non-Fungible Assets. With a dual token model, the protocol allows peers to trade a Non-Fungible Asset-Backed Token (T_{NFT}) that represent the collateralized Proofs of Consumption as Non-Fungible Asset(s) in exchange for its notional value in economy or utility tokens (T_{NFXC}). Upon the creation of the Non-Fungible Token, T_{NFT}, it can be either be held by the counterparty until the repayment remittance or re-traded for faster liquidity. The protocol's main objective is to enable immediate liquidity upon a record of the asset's Proof of Consumption on the network.

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 assets also have the challenge of illiquidity — they cannot be easily traded, locking up asset owners' capital.

This challenge is doubly extended to non-fungible assets that historically have never had access to efficient markets due to challenges arising from their non-fungible nature. The DALP overcomes these challenges by facilitating the creation of orderly, decentralized and efficient markets for non-fungible assets. High-yield non-fungible assets to be traded on these new markets include licenses, copyrights, patents, accounts receivable and other financial instruments.

2) Proof of Consumption

Any consumption event on an asset that results in an economic incentive (fiat or otherwise) to the asset owner is considered as a 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 incentive models, distinguished by how the transactions are converted to incentives.

2.1 Continuous Incentive Model

The Continuous Incentive Model assumes that the incentives paid out are a function of 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 incentives.

2.2 Fixed Incentive Model

In the Fixed Incentive Model, a digital asset 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 is either the contract or the license itself.

2.3 Structure

The composition of a Proof of Consumption is a highly generic model. Various data compositions from different source providers are normalized so that they can be denoted with a notional value. The structure of a Proof of Consumption is as follows:

- Timestamp
- Source
- Asset Type
- Metric Type
- Units
- Price Per Unit
- Currency
- Metadata

Timestamp	Source	Asset Type	Metric Type	Units	Price	Currency	Metadata

July 10th	Spotify	Music	Stream	100	0.004	USD	Subscription, Territory, Duration
July 11th	YouTube	Video	View	1000	0.0007	GBP	Subscription, Territory, Duration
July 12th	iTunes Store	Apps	Download	10	0.70	EUR	Territory
July 12th	Kindle Store	eBooks	Download	1	9.99	USD	Territory

Table 2.3.a: An example of Proofs of Consumption in the modern music industry

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 transactions data sets into NFTs while also transparently accounting for the associated incentives to the asset-holders. Investors 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 EIP721 based non-fungible token to represent the Non-Fungible Asset(s) that are the Proofs of Consumption
- **T_{NFXC}** - An EIP20 based economy 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. All incentives of the network are paid out in T_{NFXC}. All services and products built with the protocol can be accessed using T_{NFXC}.

3.1 Liquidity Network

The Liquidity Network in the implementation of the Digital Asset Liquidity Protocol where different agents participate in the execution of a trade in exchange for pre-programmed network incentives.

3.1.1 Makers

Makers are identified as the network agents that own the digital assets and their Proofs of Consumption. It is the Maker that first publishes the intent for trade in exchange for liquidity on the network. With the assistance of the relayer, Makers set the various terms for placing a Liquidity Order.

3.1.2 Takers

Takers are the agents that are responsible for supplying liquidity in the network. They fill orders published by the makers and assume the temporary ownership of the Non-Fungible Tokens i.e., the Liquidity Tokens. Takers' incentives are in making a return on the investment for providing liquidity on a published order.

3.1.3 Suppliers

The Suppliers are the network agents responsible for delivering Proofs of Consumption data sets into the Liquidity Network. They are typically the service providers of the Makers, with whom they have a Service Level Agreement(SLA) that includes two main responsibilities

1. Periodic economic payouts to Makers for ongoing consumption of their Assets
2. Delivering the Proofs of Consumption on behalf of the Maker

The delivery of Proofs of Consumption is dependent on whether the supplier is an off-chain entity or an on-chain entity. From Off-Chain suppliers, the process might be straightforward

sourcing it in the form of flat files and through API endpoints. However, the on-chain supplier would need to leverage an on-chain oracle.

Suppliers are incentivized (one-time) for their participation in the Liquidity Network by delivering the Proofs of Consumption.

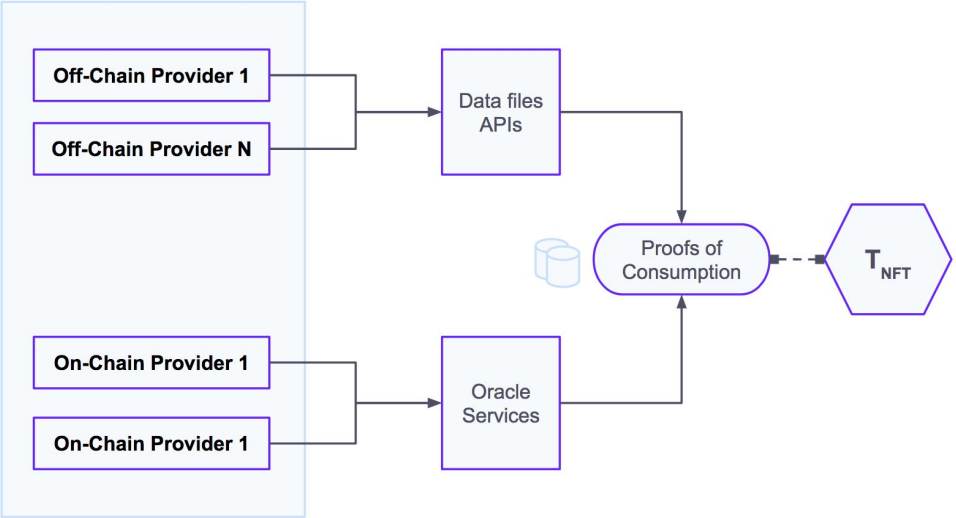

Figure 3.1.3.a: Visualization of sourcing Proofs of Consumption from off-chain and on-chain suppliers

Suppliers are the *trusted* entities that deliver the eventual remittance to the owner of the Non-Fungible Asset i.e., the Proofs of Consumption on behalf of the Maker.

Some examples of Suppliers are businesses that operate as Video Hosting Service, Music Streaming Platform, E-Commerce Website, Gaming App Store, E-Book Store, PodCast Station etc. Amazon would be a supplier delivering the sales of products and YouTube a supplier delivering reports of monetized content views.

3.1.4 Relayers

Relayers play an integral part in the Liquidity Network by aggregating Liquidity Orders issued by the Makers with the retail investors on the network. The order book is maintained in a centralized manner that provide the Takers the ability to browse through and fill orders. The

filled order is then published on the Ethereum blockchain through Smart Contracts. An agreed upon fee is collected from the Taker to facilitate the trade.

The other key function a Relayer plays on the network is to source the Proofs of Consumption from the Suppliers and Tokenize them to be represented as a Non-Fungible Liquidity Token. Relayers have the opportunity to leverage a business model by collection subscription fees in economy tokens TNFXC for these value added services.

The other key function a Relayer plays on the network is to source the Proofs of Consumption from the Suppliers and Tokenize them to be represented as a Non-Fungible Liquidity Token. Relayers have the opportunity to leverage a business model by collection subscription fees in economy tokens TNFXC for these value added services.

Actors	Network Value Add	Incentive(s)
Makers	Sells tokenized asset as orders $T_{NFT}(O_i)$	Gains liquidity in T_{NFXC} for sale of T_{NFT}
Takers	Liquidates orders by purchasing $T_{NFT}(O_i)$	Discounted purchase earning T_{NFXC} returns Resell T_{NFT} in secondary markets
Suppliers	Supplies Proofs of Consumption. Payment remittance	Earn T_{NFXC} for supplying data
Relayers	Tokenizing assets Order settlement	Earn T_{NFXC} in transaction fees

Table 3.1.4.a: 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 (Maker) and a Taker 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:

- Taker and Maker agree on Order terms (For example amount, maturity date, expiration date)
- Taker's funds are sent to Maker.
- Maker receives funds.
- Taker assumes ownership of NFT as an IOU.
- In due time, remittance payment is made by the Service Provider.
- Funds are redirected to the Taker.
- NFT is burned and the Liquidity Transaction is completed.

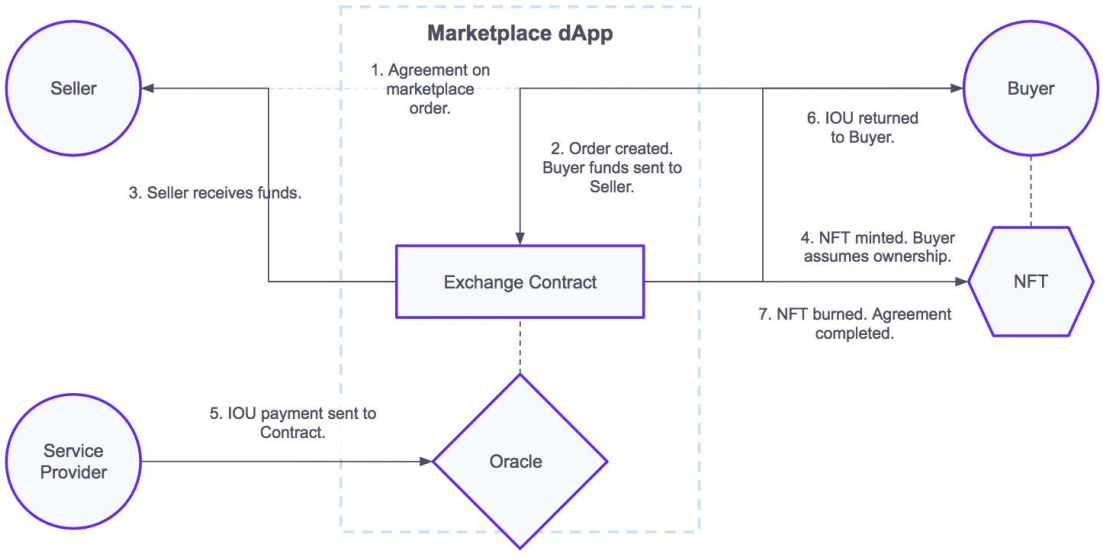

Figure 3.2.a: Example of an order execution sequence between two peers.

2.4 Secondary Markets

In the second scenario shown below, the initial Taker 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:

- Taker and Maker agree on Order terms (For example amount, maturity date, expiration date)
- Taker's funds are sent to Maker.
- Maker receives funds.
- Taker assumes ownership of NFT as an IOU.
- Taker and Secondary Taker reach agreement on re-sale of NFT.
- Secondary Taker funds are transferred to Buyer, in exchange for transfer of the NFT to the Secondary Buyer.
- In due time, remittance payment is made by the Service Provider.
- Funds are redirected to the Secondary Taker.
- NFT is burned and the Liquidity Transaction is completed.

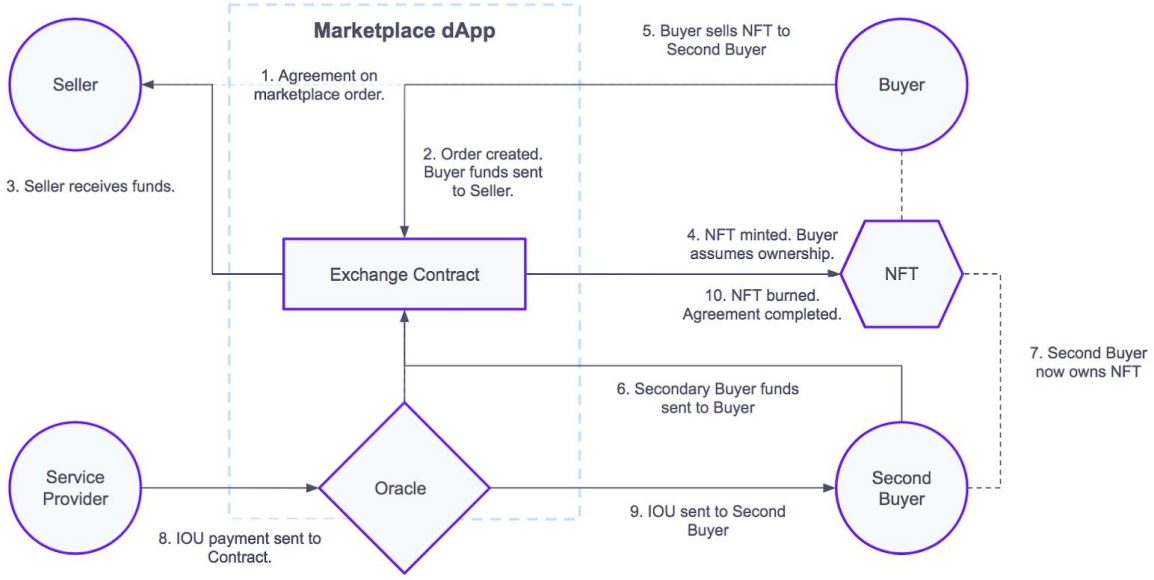

Figure 3.3.a: Example of an order execution sequence with a secondary taker scenario.

4) Specification

This section captures the specification of the off-chain relayer API that helps with order discovery among the peers and the mechanics of the Non-Fungible Exchange Smart Contract that interfaces with the Ethereum Blockchain.

4.1 Relayer Order API

This API is hosted off-chain by the Relayer to maintain its order books. All endpoint requests assume that the Content-Type: application/json header is passed.

4.1.1 POST /api/:version/orders

Creates and publishes a new order on the Relayer. The payload captures the Maker's signed intent to trade for the specified set of terms. On success, the endpoint would return an order Id. The order is signed with the Maker's Elliptic Curve Digital Signature Algorithm (ECDSA), which can be used to validate the origin and integrity of the order.

```
// REQUEST PAYLOAD
{
  "makerAddress": "",
  "makerAmount": 0,
  "takerReturn": 0,
  "relayerFee": 0,
  "expiration": "",
  "repaymentDue": "",
  "nonce": 0,
  "v": "",
  "r": "",
  "s": ""
}

// RESPONSE
{
  "orderId": 0
}
```

4.1.2 POST /api/:version/orders/:orderId/cancel

Called by the Maker to cancel the published order on the Relayer's order book

```
// REQUEST PAYLOAD
{
  "orderId": 0
}

// RESPONSE
{
  "success": true
}
```

4.1.3 GET /api/:version/orders/:orderId

Fetches the order for the given orderId.

```
// RESPONSE
{
  "makerAddress": "",
  "makerAmount": 0,
  "takerAddress": "",
  "takerAmount": 0,
  "relayerFee": 0
  "takerReturn": 0,
  "expiration": "",
  "repaymentDue": "",
  "nonce": 0,
  "v": "",
  "r": "",
  "s": ""
}
```

4.2 Non-Fungible Exchange Contract

The Non-Fungible Exchange Contract (NFXC) is the main smart contract that a Relayer interacts with to publish a filled signed order for the agreed upon terms amongst the Maker and the Taker. The contract also houses the functionality to trigger the repayment on the Liquidity Token from the supplier to the buyer.

Upon deployment of the Exchange Contract, the contract addresses capturing the Relayer wallet for fees, the economy token T_{NFXC} used for trade and the Liquidity Token T_{NFT} are set.

4.2.1 Liquidity Order

A Liquidity Order is the atomic unit of trade that captures the Maker's contract terms and the Taker's intent to fill.
An order has the following characteristics:

Name	Data Type	Description
_makerAddress	`address`	The Address of the Maker that issued the order
_makerAmount	`unint256`	The principal amount in T_{NFXC} that is being requested by the Maker
_takerAddress	`address`	The Address of the Take that has filled the order
_takerAmount	`unint256`	Total units in T_{NFXC} that the taker is transferring (i.e, Principal - Taker Fee)
_relayerFee	`unint256`	Total units in T_{NFXC} that the taker will incur as Fees to the Relayer
_takerReturn	`unint256`	Total units in T_{NFXC} that the taker will make as return on repayment
_repaymentDue	`unint256`	Unix timestamp indicating when the repayment will be made to the Taker from the Supplier
_economyToken	`address`	The Address of the EIP20 Token T_{NFXC} which is the main Economy Token
_liquidityToken	`address`	The Address of the EIP721 Token T_{NFT} which is the Non-Fungible Token
_nonce	`uint256`	An integer value used to differentiate the hashes of the Liquidity Orders with identical parameters
_v	`uint8`	Recovery Id of the Maker's ECDSA signature

_r	`bytes32`	32 byte output of the Maker's ECDSA signature
_s	`bytes32`	32 byte output of the Maker's ECDSA signature

4.2.2 Contract Interface

```
interface IExchange {
    function setRelayerWallet(
        address _paperchainWalletAddress
    ) external;

    function setEconomyTokenAddress(
        address _tokenAddress
    ) external;

    function setOrderTokenAddress(
        address _tokenAddress
    ) external;

    function getOrderInfo(
        uint _orderId
    ) external view returns (bytes32, uint256, address);

    function fillOrder(
        uint _orderId,
        address _makerAddress,
        uint256 _makerAmount,
        address _takerAddress,
        uint256 _takerAmount,
        uint256 _relayerFee,
        uint256 _takerReturn,
        uint256 _remittanceDueEpoch,
        uint8 _v,
        bytes32 _r,
        bytes32 _s
    ) public returns (uint256);

    function remitOrder(
        uint _orderId
    ) public;

    function tokenFallback(
        address _from,
        uint256 _value,
        bytes _data
    ) public;
```

```
    function withdrawToken(
        address _to,
        address _tokenAddress
    ) public;
}
```

4.3 Order Lifecycle

4.3.1 Order Publishing & Discovery

Makers publish their signed intent to trade on the Relayer's off-chain order books which makes it available for discovery for Takers browsing through. Order terms are set by the Maker and negotiated with the Taker. Since the Relayer is sourcing Proofs of Consumption from the supplier, `repaymentDueDate` is worked out in conjunction with the Supplier and served to Maker during the publishing phase. For the network services, Relayer specifies a `relayerFee` which is a deduction from the Maker's principal amount. Maker specifies the `returnFee` as the amount from the principal that they are willing to offer the Taker as return for filling the order.

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

- Principal amount of 10,000 T_{NFXC}
- Relayer Fee of 50 T_{NFXC} (0.5% of 10,000 set by Oracle during deployment)
- Taker return of 250 T_{NFXC} (2.5% of 10,000 set by Oracle during deployment)
- Which makes the final liquidity amount to be 9,700 T_{NFXC} (Principal amount - (Relayer Fee + Taker Return))

Should the order be filled by a Taker, Alice receives 97% of the original posted amount.

4.3.2 Order Fills

Takers browse through the order books of the Relayer to discover and fill the available orders from Makers. While this is facilitate by the Relayer Order API, the action of filling the order is executed through the Smart Contract method `fillOrder()`. The Smart Contract is designed to validate the Maker's identity through the ECDSA signature supplied in the request. Upon filling, the Smart Contract makes 3 token transfers

- Transferring EIP20 economy tokens T_{NFXC} that is the Relayer Fee to the Relayer Wallet
- Transferring EIP20 economy tokens T_{NFXC} that is the Liquidity Amount to the Maker
- Minting and Transferring 1 EIP721 token T_{NFT} that encompasses the order to the Taker

The taker, 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 `repaymentDueDate` and he is happy to take that risk for a short term return.

When Bob fills the order, Bob's wallet has now been debited with 9750 EIP20 token T_{NFXC} and credited with 1 newly minted EIP721 token T_{NFT}. The T_{NFT} is the EIP721 token that is tied to the order that is indicative of Bob's ownership of the order. The $T_{NFXC}(O_i)$ is the IOU from Alice to Bob.

From the 9750 T_{NFXC}, 50 T_{NFXC} is transferred to the Relayers's wallet address as 0.5% relayer fee and the remaining 9700 T_{NFXC} is transferred to Alice the maker as the Liquidity Amount.

4.3.3 Repayment Remittance

Relayer and/or the Supplier will be authorized off-chain, by the maker Alice, to remit funds to Bob when her funds become available by the previously mentioned `repaymentDueDate`. The repayment remittance is accomplished using the `remitOrder()` function on the Smart Contract.

10,000 T_{NFXC} are transferred from the Supplier to Bob's wallet. Bob, now having received the funds, has made a return of 250 T_{NFXC} (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 taker. If this is the case, the Relayer will remit the payment to the current owner of the $T_{NFT}(O_i)$.

	Order state	Alice (0xa1b2c30)	Bob (0xq1w2e3)	Relayer (0xz5x6c7)
Publish Order	*orderId* 1 *maker* 0xa1b2c30 *orderAmt* 9700 *relayerFee* 50 *takerReturn* 250 *expiration* July 28 '18 *repaymentDue* Sept 1 '18	0 T_{NFXC}	10,000 T_{NFXC}	X* T_{NFXC}
Fund Order	*status* Funded *taker* 0xq1w2e3	+ 9700 T_{NFXC}	- 9700 T_{NFXC} + 1 $T_{NFT}(O_1)$	+ 50 T_{NFXC}
Order Repayment	*status* Remitted	+ 10,000 T_{NFXC} - 1 $T_{NFT}(O_1)$	- 10,000 T_{NFXC}	

** X is the token supply in Relayer's wallet*

Table 4.3.3.a: Account state changes of an order's lifecycle on the marketplace

5) Token Utility

The economy token T_{NFXC} is a multi-purpose EIP20 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 on 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 maker from volatility if needed). Here, the idea of 'stability' is rather a Token Engineering characteristic than a different 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 mechanisms 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 protocol itself allows for some exciting extensibility per the desired marketplace configuration:

- Offering an improved level of price discovery by allowing the taker(s) to bid for a price ensuring the maker to get the best price on the market. This would imply that the maker withdraws the funds from the taker on accepting the bid
- For larger valued orders, facilitation of funding via taker pools where takers' return staked to their contribution allowing them to spread their risk
- Creating market conditions and incentivization mechanisms 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 maker'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.