paperchain

Non-Fungible Exchange Contract (NFXC) Standard

[DRAFT]

Daniel Dewar, Rahul Rumalla, David Tomaselli

Version 0.1 April 07, 2018

Abstract

Paperchain is building a decentralized protocol to standardize the way non-fungible assets can be traded between peers on a blockchain. This paper defines the Non-Fungible Exchange Contract (NFXC) protocol that facilitates a peer-to-peer methodology for exchange. In this framework, two peers (a Buyer and a Seller), transact with each other in a marketplace. The Buyer is issued a Non-Fungible Asset-Backed Token (NFT) that collateralizes the Seller's Non-Fungible Asset(s) in exchange for the current notional value of the Asset. The NFXC defines a mechanism for an IOU remittance to the Buyer, attached to the NFT. A Marketplace Oracle Service helps facilitate the IOU issuance to the Buyer, while also executing the burning of the NFT, completing the trade.

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. Media transactions are one such market.

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 transaction data, such as streams, impression or click data, are available in near real-time, but goes largely unutilized in support of core business processes, in particular liquidity.

Why tokenize non-fungible assets

ERC-721 is a recently accepted token standard[1] on the Ethereum blockchain that tokenizes non-fungible assets into non-fungible tokens (NFT). Unlike ERC-20, each token is unique in its input and characteristics and divisible by 1. Once tokenized, the non-fungible asset becomes a liquid asset, making it ideal for trading and secondary market monetization.

[1] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md

A futures model, or futures-like model, introduces constraints around token lifetime, however, this does not disallow the token from being collateralized in peer-to-peer marketplaces and lends itself to mint and burn strategies of token management.

2) The Non-Fungible Exchange Contract (NFXC) Protocol

The Non-Fungible Exchange Contract (NFXC) addresses the need for a protocol that enables peer-to-peer trading of non-fungible assets, and permits secondary market opportunities for the same assets. It is built as an ERC-223[2] compliant token contract or any token that is backward compatible to ERC-20.

The NFXC allows for decentralized trading of non-fungible assets on any platform that supports ERC-20 tokens. The NFXC protocol is comprised of four components:

1. The Non-Fungible Exchange Contract (NFXC)
2. The ERC-223 Token of NFXC (T_{NFXC})
3. The Non-Fungible Token (T_{NFT})
4. The Marketplace Oracle Service (MOS)

It is best that these services operate on separate contracts to ensure greater flexibility and security with regards to contract code updates.

[2] https://github.com/ethereum/EIPs/issues/223

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 NFXC. 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 NFXC.
6. Funds are sent to the Buyer.
7. NFT is burned and the Agreement is completed.

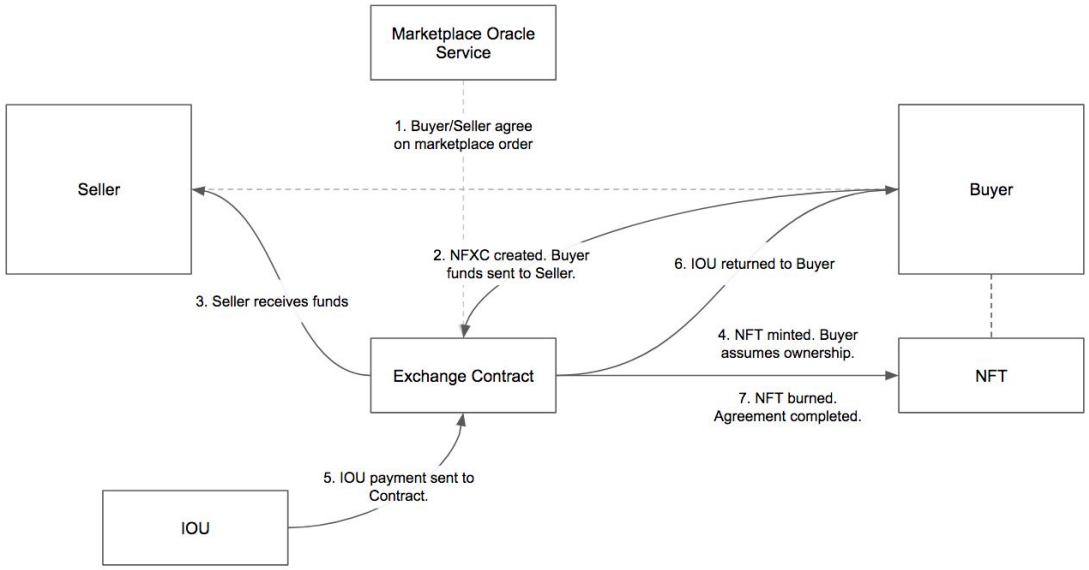

Figure 1: Example of a NFXC execution sequence between two peers.

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 NFXC.
9. Funds are sent to the Secondary Buyer.
10. NFT is burned and the Agreement is completed.

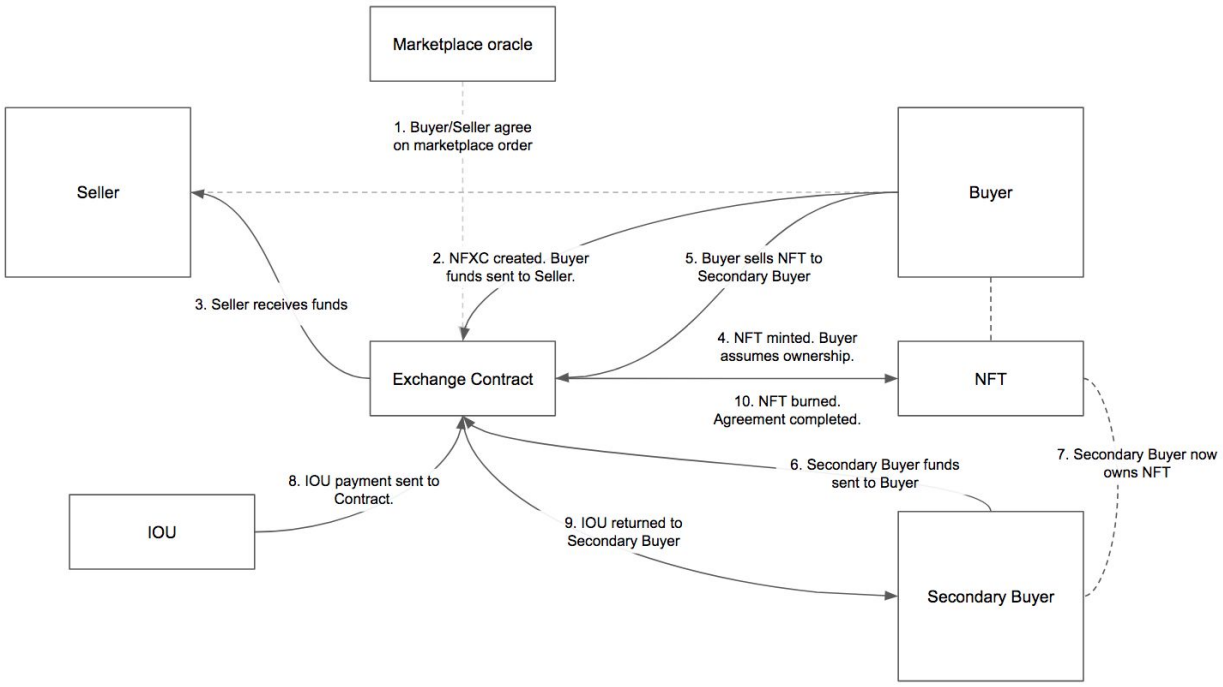

Figure 2: Example of a NFXC execution sequence with a secondary buyer scenario.

3) NFXC Token

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.

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. An order is with the following characteristics

- Order ID

  The unique index of the order
- Order Amount

  The value of the order represented in T_{NFXC} (ERC-223 tokens)
- Charge Amount

  The fraction of the total amount which the Market charges as a charge fee
- Return Amount

  The fraction of the total amount which the Buyer makes as a return
- Seller's Address

  The wallet address of the Seller
- Buyer's Address

  The wallet address of the Buyer
- Funding Deadline

  The timestamp indicating by which the seller desires the order to be funded by
- Remittance Due

  The timestamp, set by the MOS, indicating when the expected remittance is due
- Order 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
- External Fingerprint ID

  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 supplied with 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 *Bob* the *buyer* 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 i.,e

- 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)
```

Bob the **buyer** 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 fund's 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 $T_{NFT}(O_i)$ i.e., transfer his ownership to another buyer. If this is the case, the MOS will remit the payment to the current owner of the $T_{NFT}(O_i)$.

Lifecycle State Changes

	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) Non-Fungible Token (NFT)

The Non-Fungible Token is a proposed Ethereum Request for Comment Standard (ERC-721) that provides the means of tokenizing non-fungible assets. It's key characteristics are that each token is unique and divisible only by 1.

The NFT is defined by:

- Seller ID
- Asset value
- Date type

- Data characteristics
- Notional value (value is the agreed price)
- MOS ID

5) Marketplace Oracle Service (MOS)

The Marketplace Oracle Service provides the NFXC with the ability to reference the Marketplace Orderbook for matching Contract data to Marketplace data, using the External Fingerprint ID or Correlation ID.

Additionally, the MOS functions can be listed as:

- Configure and deploy markets that set the parameters such as market fee % and return % for the buyer
- Maintain an off-chain cache of the Orderbook and present the appropriate orders to buyers using endpoints
- Check and validate the uniqueness of new orders and new NFTs minted
- Periodically check for seller's account status and settle off-chain transfers from their clients
- Facilitate remittance from seller to buyer by the expected remittance date

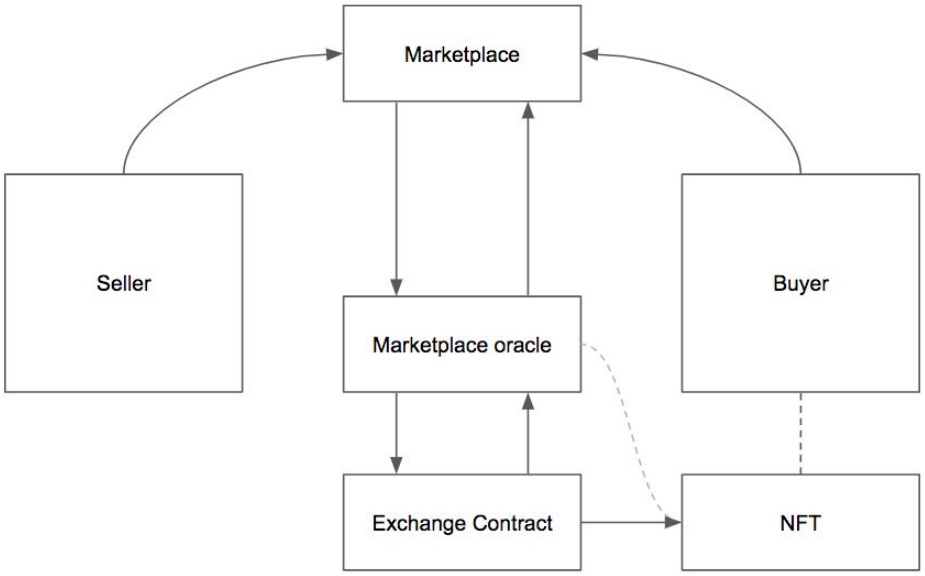

Figure 3: The NFT calls upon the Marketplace Oracle Service to populate its characteristics at the time of minting.

MOS could be considered as an excellent use-case for implementing Ethereum Foundation's new research, Plasma Sidechains[3].

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 market configuration:

[3] https://plasma.io/plasma.pdf

- 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, interest bearing loans etc
- 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.