CRASHR PROTOCOL SPECIFICATION

Reza Jhay Lacanlale, Clark Alesna

On behalf of SAIB Inc.

Crashr, a decentralized marketplace on Cardano, facilitates multi-asset trading. It extends beyond the standard ADA-NFT exchange model, enabling users to trade various digital assets, fungible and non-fungible, in a single transaction. Leveraging the established JPG Store[1] v3 contract as a foundation, this whitepaper explores the key differences between Crashr and JPG Store[1] v3, while also detailing the off-chain architecture that powers the Crashr platform.

1. Introduction

The Crashr protocol is a decentralized marketplace for digital assets on the Cardano network. It leverages the robust foundation of the open-source JPG Store[1] v3 contract, well-established for its secure and user-friendly smart contract that facilitates efficient asset trading. Notably, this contract excels at handling multiple NFT-ADA transactions.

This whitepaper explores the core functionalities inherited from the JPG Store[1] v3 contract that Crashr retains, while also unveiling the innovative features it introduces. Crashr aims to significantly improve trading on the Cardano blockchain by enabling seamless transactions involving various digital assets in a single interaction. This includes fungible tokens, NFTs, and ADA. Additionally, the paper delves into the off-chain architecture that underpins the Crashr platform and how these components facilitate interaction for both users and third-party applications.

2. Crashr Protocol Overview

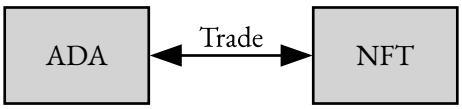
The Crashr platform operates on a single, comprehensive smart contract that governs all core functionalities, including listing assets, making bids on listings, buying assets, accepting bids, and managing listings (cancellation and updates).

- **Listing:** Users can list their digital assets on the marketplace by depositing them into the smart contract and specifying a desired price or initiating an auction.
- **Making an Offer (Bidding):** This feature allows users to submit bids on existing listings, potentially competing with other buyers. The seller can then choose to accept the highest bid or decline all offers.
- **Buying:** To purchase an asset, users initiate a transaction with the smart contract, paying the seller's price along with the marketplace and royalty fees (if applicable).

- **Accepting an Offer:** Similar to buying an asset, users can accept an offer received through the off-chain matching system. On the blockchain, this action appears identical to a regular purchase.
- **Canceling a Listing or Offer:** Users can remove their listed assets from the marketplace by initiating a cancellation transaction. This requires a digital signature to confirm ownership and authorization for withdrawing the assets.
- **Updating a Listing:** This functionality combines aspects of canceling and relisting. Users can withdraw their assets from an existing listing, modify details (e.g., price), or add different assets before putting them back up for trade on the marketplace.
- **Multiple Listings/Offers in a Single Transaction:** Crashr allows users to fulfill multiple listings or offers within a single transaction. Each listing or offer must include its individual marketplace fee. For transactions involving multiple listings/offers, an "offset" property within the redeemer specifies the position of each marketplace fee, followed by the payouts for that listing. This ensures accurate fee allocation and payouts for each fulfilled listing or offer.

2.1. Traditional Limitations

Current Cardano marketplaces often adopt a limited trading model, primarily facilitating exchanges between ADA and NFTs. While this approach has served as a solid foundation for basic transactions, it falls short of addressing the evolving needs of the digital asset community.



2.2. What Crashr Protocol Offers

The Crashr protocol brings a feature that broadens trading options, enabling transactions involving various combinations of assets in one go. This includes exchanging ADA for NFTs, NFTs with each other, or more complex trades that involve ADA, NFTs, and several fungible tokens. This capability extends the range of possible transactions on the Cardano blockchain, allowing for a more versatile trading landscape.

3. The Marketplace Smart Contract

The Crashr protocol maintains the core functionalities of the JPG Store[1] v3 contract, with modifications to the validation logic to accommodate multi-asset trades. This section outlines the key features of the original contract and details the enhancements introduced by Crashr.

3.1. Listings and Offers

One of the key differences between JPG Store[1] v3 and Crashr is that JPG Store[1] uses different contracts or validators for listing and offer transactions. In Crashr, we use the same contract for both of these actions. This consolidation requires an off-chain mechanism to determine whether a transaction is an offer or a listing, since the on-chain data doesn't distinguish between the two.

This change simplifies how users interact with the platform.

3.2. Payouts

We've updated the shape of the payouts to accommodate multi-asset trading. Payouts are a list of addresses and assets that the user and royalty recepients will receive. The marketplace payout is not included in the list because it is enforced by the contract and is always the first output in each listing being fulfilled. Payout value is a map of assets and quantities, where quantities are only positive non-zero integers. If the quantity is zero or negative, buying the asset will always fail but the owner can still cancel the listing.

3.2.1. JPG Store v3 Payout Structure

The JPG Store[1] payout structure allows the user to specify their address and the amount of ADA they want to receive for the trade:

```
pub type Payout {
  address: Address,
  amount: Int
}
```

3.2.2. Crashr Payout Structure

In the Crashr protocol, we used `Value` instead of `Int` to allow the user to specify multiple assets in the payout. The `Value` type contains tokens indexed by PolicyId and AssetName:

```
pub type Payout {
  address: Address,
  amount: Value
}
```

3.2.3. Payouts Example

To illustrate how payouts work, imagine a scenario where a user wishes to exchange 10 ADA and NFT_1 for NFT_2. In this example, the user would transfer the specified assets to the marketplace contract, accompanied by the following datum:

```
{
  "payouts": [
    {
      "address": "seller_wallet_address",
      "value": {
        "policy_id_of_NFT_2": {
          "NFT_2": 1
        }
      }
    },
  ],
  "owner": "owner_pkh"
}
```

In this scenario, the user's goal is to receive NFT_2 in exchange. The policy_id_of_NFT_2 represents the policy ID associated with the NFT_2 asset, while NFT_2 itself is the name of the asset. The owner_pkh refers to the public key hash of the user who has placed the assets into the marketplace contract.

This particular trade does not include a royalty payout. However, should a royalty payout be applicable, it would be detailed in the list of payouts, specifying both the address of the NFT's original creator and the ADA amount designated as the royalty.

3.2.4. Collection Offer Payout Example

There are instances where a user might be interested in acquiring any asset from a specific collection, often referred to as a collection offer. The Crashr protocol accommodates this type of transaction. Here, the user is required to specify just the policy ID of the desired collection along with the quantity of assets they wish to receive. A buyer can meet this request by transferring any asset governed by the specified policy ID. The structure of such payouts is outlined as follows:

```
{
  "payouts": [
    {
      "address": "seller_wallet_address",
      "value": {
        "policy_id_collection": {
          "": 1
        }
      }
    },
  ],
  "owner": "owner_pkh"
}
```

3.3. Marketplace Fee

The marketplace fee, mandated by the protocol, is designed to fund the ongoing maintenance and development of the marketplace. This fee is predefined within the contract's code and is consistently positioned as the first output when fulfilling a listing.

3.3.1. JPG Store v3 Fee Calculation

Within the JPG Store[1] v3 contract, the marketplace fee is determined to be 2% of the overall payout sum. Given that the original contract was designed exclusively for NFT-ADA exchanges, the process for calculating this fee remains relatively simple.

$$\text{Total Fee} = \left(\sum_{i=1}^{n} P_i\right) * \left(\frac{2}{100}\right)$$

Where:

- **Total Fee** is the total marketplace fee.
- $\mathbf{P_i}$ is the payout amount for each payout.
- **n** is the number of payouts.
- **2/100** is the 2% fee rate.

3.3.2. Crashr Fee Calculation

With the Crashr protocol supporting multi-asset arbitrary trades, the marketplace fee structure has been updated to cater to the wider array of trading options. The fees for ADA transactions continue as per the original contract's terms.

3.3.3. Unique Token Fee

The addition of multi-asset trades led to the introduction of a unique token fee. This fee, set at 1 ADA for each distinct asset requested by the listing owner, is predefined in the contract and automatically applied. For fungible token transactions involving less than 100 tokens, this unique token fee applies. When the quantity reaches 100 or more, the fee adjusts to match the original marketplace's 2% of the total token payout.

$$\text{Total ADA Fee} = \max\left(\sum_{\text{i=1}}^{u}(Q_i < 100 ? 1 : 0) * U_f + \sum_{\text{k=1}}^{n} P_k * \left(\frac{2}{100}\right), 1\right)$$

Where:

- **Total ADA Fee** is the part of the fee related to ADA transactions and unique assets.
- **u** is the total number of unique assets.
- $\mathbf{Q_i}$ is the quantity of the i-th unique asset.
- **U_f** is the ADA fee per unique token, applied to unique assets with quantities less than 100.
- **n** is the total number of ADA payouts.
- $\mathbf{P_k}$ is the k-th ADA payout amount.

The first summation calculates the unique token fees for assets with quantities less than 100 and adds the 2% fee for each ADA payout. The max function ensures this part of the fee is at least 1 ADA.

$$\text{Total Token Fees} = \sum_{\text{j=1}}^{m}\left(Q_j \geq 100 ? F_j * \left(\frac{2}{100}\right) : 0\right)$$

Where:

- **Total Token Fees** is the part of the fee related to fungible token transactions where the quantity exceeds 100.
- **m** is the total number of fungible token transactions.
- $\mathbf{Q_j}$ is the quantity of the j-th fungible token transaction.
- $\mathbf{F_j}$ is the payout amount for the j-th fungible token transaction.

The second summation applies a 2% fee to each fungible token payout where the quantity is 100 or more.

$$\text{Total Fees} = \text{Total ADA Fee} + \text{Total Token Fees}$$

Where:

- **Total Fees** is the total fee calculated by the Crashr protocol, combining the ADA-related fees and the fees from fungible token transactions.

3.4. Royalties

Royalty is an important part of the NFT ecosystem, enabling creators to receive a portion of the sale price each time their NFT is traded. The JPG Store[1] v3 contract handles royalty calculations directly, applying them as a percentage of the total listing amount in NFT-ADA trades.

The Crashr protocol also supports royalties, allowing users to allocate a royalty fee for the NFT's original creator. This fee, usually in ADA, is determined offchain based on the asset's estimated value at the time of listing. Although the Crashr contract doesn't mandate the inclusion of this fee,

it provides the option to add it to the list of payouts, with a set minimum of 1 ADA for the royalty fee.

Note:
With multi-asset trades, the calculation of royalty fees is significantly influenced by the specific assets involved. When trades include assets from various projects, it's possible to list each project's royalty address in the payouts, as long as the assigned ADA value adheres to the minimum threshold of 1 ADA.

3.5. Revenue Sharing

To incentivize user engagement and trading activity, our platform implements a streamlined revenue-sharing model, distributing generated income across three fundamental components of the protocol. The treasury, staking, and buyback.

3.5.1. Treasury

The treasury supports the platform's development, maintenance, and expansion. It's also staked to the Crashr Stake Pool to aid in decentralization and enhance the network. To maintain heightened security and trust, the treasury's funds are secured in a multi-signature wallet.

Allocation: 60% of all collected fees.

3.5.2. Staking

Staking rewards play a crucial role in promoting long-term engagement and stability within the ecosystem. Rewards are determined based on the amount and duration of $CRASH tokens staked, with the total fees allocated for staking distributed to participants in $CRASH tokens.

Allocation: 20% of all collected fees converted to $CRASH tokens.

3.5.3. Buyback

Aimed at stabilizing and boosting the $CRASH token's value, the buyback is conducted quarterly, it uses a set portion of the fees to repurchase $CRASH tokens. These tokens are then reintegrated into the treasury, earmarked for release via DAO approval or towards the protocol rewards.

Allocation: 20% of all collected fees plus all ADA generated from staking rewards.

3.6. Security

Security is a critical aspect of any decentralized marketplace, and the Crashr protocol upholds the rigorous security standards set by the JPG Store[1] v3 contract. This commitment ensures the safeguarding of assets and the precise execution of transactions.

With the adaptation of the contract to facilitate multi-asset trades, additional security measures have been implemented. These modifications are currently undergoing thorough audits by security experts to verify that the protocol's integrity and security remain uncompromised. The Crashr marketplace contract will be audited by Sundae Labs[3].

4. Marketplace Transaction Flow

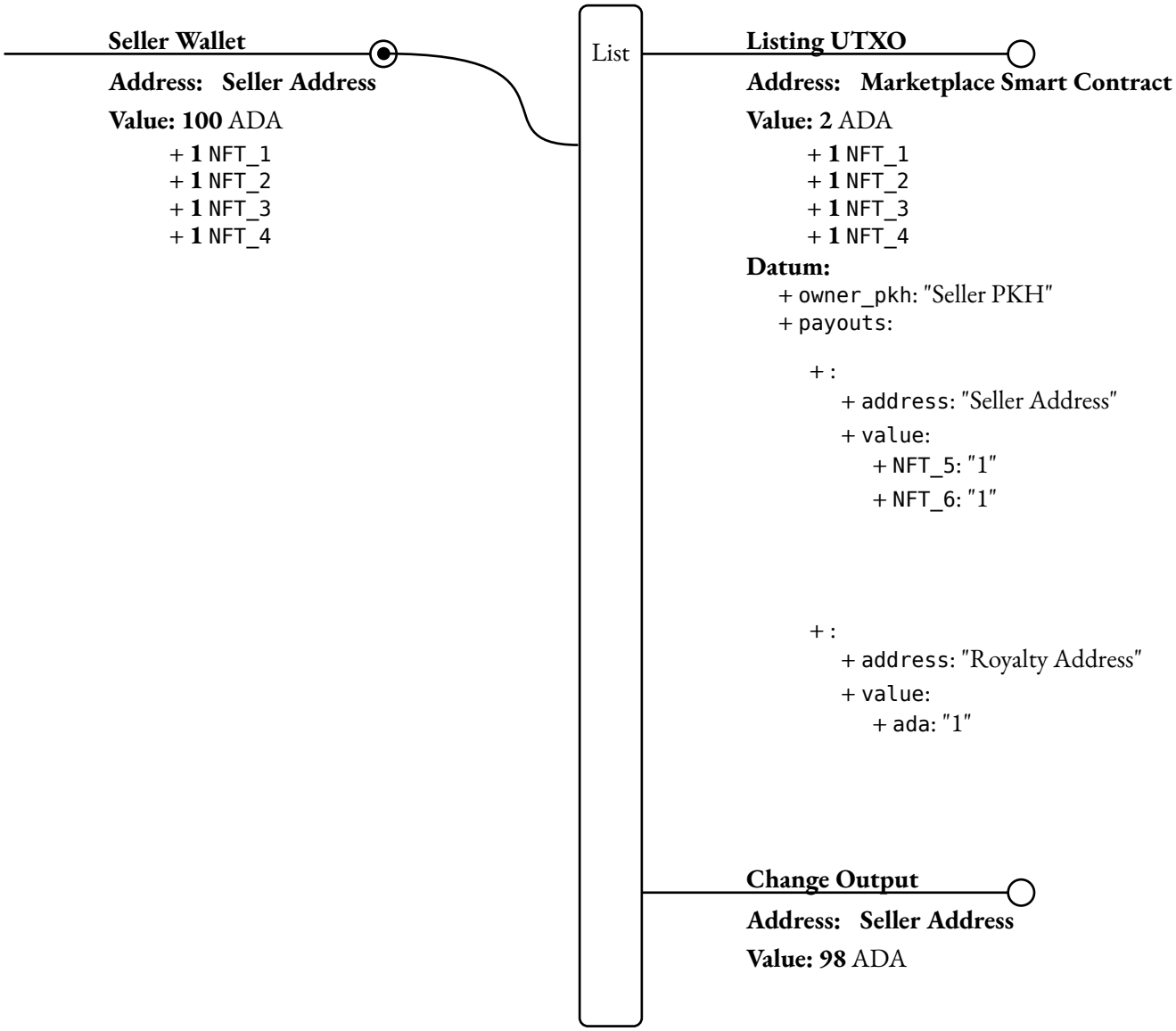
To grasp the functionalities enabled by the Crashr protocol, it's useful to explore the transaction flow for essential actions such as Listing, Buying, and Canceling/Updating a listing. This exploration will provide insight into how transactions are structured within the Cardano network. These key operations form the backbone of the marketplace, facilitating a diverse array of asset trades and enhancing user interaction with the platform.

4.1. Creating a Listing or Offer

The process for listing assets under the modified contract remains largely aligned with the practices established by JPG Store[1]. The notable enhancement is the contract's expanded capability to accommodate multiple assets, broadening the scope of transactions beyond the original single-asset framework.

4.1.1. Listing Requirements

1. **User** must **send the assets they wish to trade** to the **marketplace contract**.
2. **User** must **specify the assets he wish to receive in return** in the **output datum** and optional royalty payouts.
3. For transactions exclusively involving **native assets**, users are required to **lock the minimum ADA** required by the protocol along with the assets.
4. Optionally, **users can designate a royalty fee** for the **original NFT creator**. This fee, calculated offchain and payable solely in ADA, is based on the NFT's **estimated value at listing**. In cases of trades involving **multiple NFTs from distinct projects**, each project's **royalty address** may be included in the payout, provided the ADA value meets the **minimum requirement of 1 ADA**.



Note: Sample *listing transaction* where the owner locks 4 NFTs and 1.5 ADA in the marketplace validator address with a datum specifying the seller's address and the assets they want to receive in exchange and optional royalty payouts. The seller has the option to request a non-specific asset from a particular collection. To do this, within the value property of the payout, the seller needs only to provide the policy ID. The buyer can fulfill this request by sending any asset that falls under the given policy ID. Furthermore, the seller is able to request any quantity of assets under a policy ID without needing to name the specific assets. This functionality aligns with the collection offer feature of the JPG Store[1] contract.

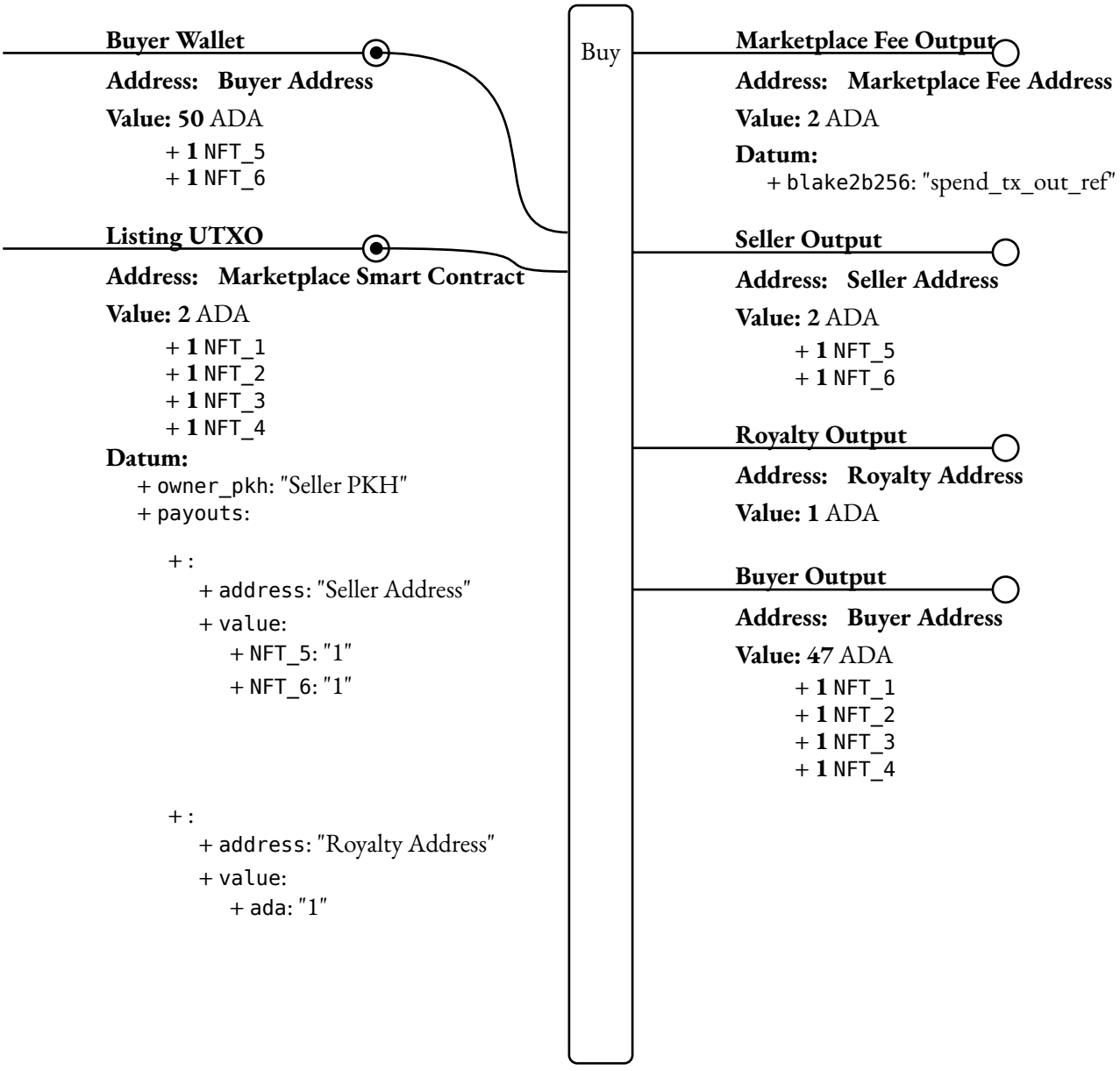
4.2. Buying or Accepting an Offer

To successfully complete a transaction, buyers are required to transfer the assets specified by the seller to the seller's address. The contract also requires buyers to send a 2% fee to the marketplace, based on the total of the seller's payout plus any royalty fees. The 2% fee applies to ADA (2% or 1 ADA, whichever is higher) and to fungible tokens transactions that reach the threshold of 100 tokens.

Given the contract's capability for multiple asset trades, there are instances where only minimal ADA is required. To accommodate this, the contract has been adjusted to include a unique token fee. The unique token fee necessitates buyers to send 1 ADA for each unique asset requested by the seller. This fee also applies to fungible token transactions that involve quantities below the 100-token threshold.

4.2.1. Buying Requirements

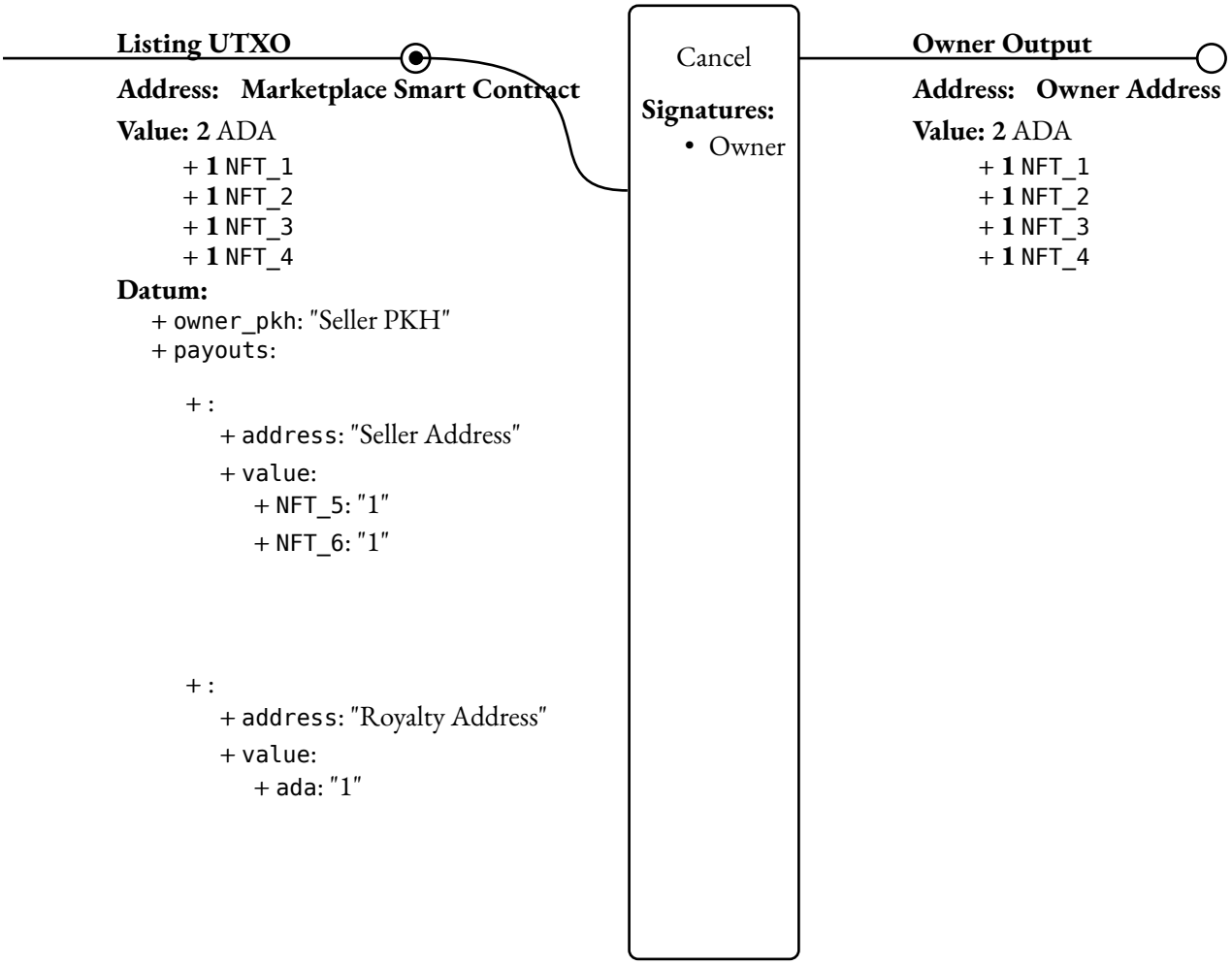
1. **Buyer must send the marketplace fee to the marketplace fee address** hardcoded in the contract. It is enforced that the **first payout must be the marketplace fee**; subsequent payouts can be in any order. The marketplace payout also needs to be tagged with the **blake2b256 hash of the spend_tx_out_ref** to prevent double satisfaction.
2. **The buy redeemer has an offset property**, an optimization from the original JPG Store[1] contract. It indicates the **current payout index being processed on-chain**.
3. **Buyer must send the assets specified by the seller to the seller's address.**
4. If a **royalty fee** is specified, the **buyer must send the royalty fee to the original creator of the NFT**. This fee, calculated offchain, is not enforced by the contract but is expected to be in ADA, based on the **estimated price of the assets** at the time of listing.



Note: Sample *buying transaction* where the buyer satisfies the seller's requirements by sending the requested assets to the sender's address, the buyer must also satisfy royalty payout requirements if applicable and send the 2% fee + 1 ADA for each unique asset requested. Minimum marketplace fee is 1 ada.

4.3. Cancel or Update a Listing

The contract enables the listing owner to cancel or update their listing, retaining the same process as established by the JPG Store[1] contract. The only requirement for these operations is the owner's signature on the transaction.

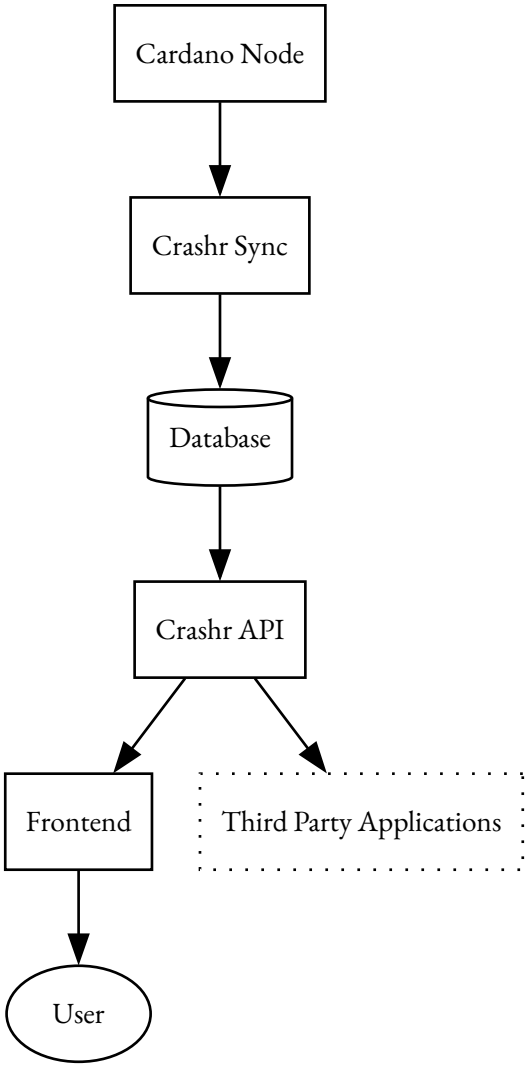


Note: Sample *cancel transaction* where the owner cancels the listing and retrieves the locked assets.

5. Offchain Architecture

The offchain architecture of the Crashr platform is designed to support the main functionalities of the marketplace, making trading operations seamless and efficient. Offchain components are essential for enhancing user experience, providing real-time data processing, and allowing for complex multi-asset trades. All offchain parts are developed in C# using the .NET Framework.

5.1. High Level Overview of Offchain Components



5.2. Crashr Sync

Crashr Sync is an essential component of the offchain architecture, designed to process data from the Cardano blockchain in real-time and record it block by block into a database. This ensures the marketplace is consistently synchronized with the latest blockchain activities, providing users with up-to-date information for their trading activities. Built upon the technologies of Cardano Sync by SAIB Inc. and the Pallas library by TxPipe[2], Crashr Sync facilitates efficient blockchain data processing and indexing.

The component takes cues from the Scrolls project by TxPipe[2], offering a framework for effective blockchain data indexing, which aids in improved data querying and management. Crashr Sync utilizes targeted reducers to index specific data types important for marketplace operations, such as smart contract interactions, asset metadata, and token pricing, keeping the platform responsive and current.

Note:
Crashr Sync uses the Cardano Node to read blockchain data.

5.2.1. Reducers in Crashr Sync

Crashr Sync is composed of several indexers called **reducers**, each designed to handle specific types of blockchain data. The key reducers in Crashr Sync include:

- **ListingByAddress Reducer:** Indexes smart contract activities for each wallet, capturing detailed transaction data including listings, offers, and trade details, along with transaction hashes and buyer addresses.
- **ListingByAsset Reducer:** Focuses on indexing information related to individual assets, using policy ID and asset name as primary keys for efficient querying and management of asset-related data.
- **MetadataByNft Reducer:** Ensures the marketplace reflects the latest NFT metadata by tracking updates, thereby maintaining a current and comprehensive database of NFT details.
- **Nft Reducer:** Monitors the lifecycle of NFTs, including minting and burning transactions, to provide an accurate real-time overview of NFT availability on the blockchain.
- **TokenPrice Reducer:** Sources fungible token prices from DEX liquidity pools to facilitate accurate ADA value calculations for tokens at any given moment, essential for multi-asset trades within the marketplace.

5.3. Crashr API

Crashr API is a component designed to expose blockchain data in a more digestible format. This API layer serves as the bridge between the complex data indexed by Crashr Sync and the user-facing elements of the marketplace, ensuring that information is accessible, understandable, and actionable. Furthermore, the Crashr API opens avenues for businesses looking to leverage our platform's rich data ecosystem, offering them the tools to build innovative applications on top of this data.

5.3.1. API Modules

The Crashr API is structured into three primary modules each tailored to serve distinct data sets and functionalities relevant to marketplace operations and third-party integrations. The core modules include:

- **Marketplace Module:** This module is the backbone of the marketplace's transactional interface, offering a comprehensive collection of endpoints that cover the entirety of marketplace activities. From listings and offers to completed trades, this module ensures that users and third-party applications have real-time access to transactional data.
- **NFT Module:** Dedicated to the nuanced needs of NFT interactions, this module provides endpoints that delve into NFTs and their associated metadata. This module facilitates easy access to NFT details, including ownership history, metadata changes, and current listings, making it an invaluable resource for both marketplace users and NFT-centric applications.
- **Price Module:** Recognizing the importance of accurate and timely pricing information in a dynamic marketplace, this module aggregates and serves data related to token prices. This includes real-time price feeds, historical price data, and analytical insights derived from market trends.

Note:
More modules can be added as the platform evolves and new features are introduced.

5.3.2. API for Third Party Applications

The Crashr API is open to businesses and developers looking to integrate with our platform, offering a suite of endpoints that provide access to blockchain data, marketplace activities, and NFT metadata.

5.4. Crashr UI

The Crashr UI is the user-facing component of the marketplace, designed to provide an intuitive and engaging experience for users interacting with the platform. The UI is built on modern web technologies, ensuring a seamless and responsive interface that caters to a diverse range of users. The Crashr UI is optimized for both desktop and mobile devices, offering a consistent experience across various platforms.

6. Future Development

While we are satisfied with the initial core functionalities of the Crashr protocol, we recognize that there is still room for growth and improvement. Our team is committed to ongoing development and enhancement of the platform, with a focus on expanding the marketplace's capabilities and improving the user experience.

We are also exploring opportunities to expand the Crashr ecosystem. Some of our plans for the future include:

- **SDKs:** Developing API SDKs and Embeddable SDKs to enable seamless integration with third-party applications and services.
- **CIP-68 Identity/Profile Tokens:** Exploring the implementation of CIP-68 Identity/Profile Tokens to enhance profile management within the marketplace.
- **Marketplace Aggregator:** Building a marketplace aggregator to provide users with a comprehensive view of all available listings and offers across multiple platforms.
- **Staking Platform:** We'd like to integrate staking capabilities into the platform to enable users to earn rewards by staking their assets.
- **Cross-chain compatibility:** Exploring interoperability with other blockchains to enable multi-chain asset trading. This includes cross-chain tradings, bridging assets, and more.
- **Real-World Asset Trading:** We're also exploring tokenizing real-world assets and enabling their trading on the Crashr platform.

7. Acknowledgements

Although we take great pride in the development of this protocol, it's important to acknowledge that our progress was significantly enhanced by the contributions from the Cardano open-source community. We extend our gratitude to the following projects for their invaluable contribution:

- **JPG Store**: For the foundational contract that Crashr is built upon.
- **TxPipe**: For providing open-source tools that make building on Cardano easier.
- **CardanoSharp** & **Orion**: For their open-source Cardano Cryptographic and Serialization library for .NET applications.
- **Sundae Labs**: For the smart contract audit.
- and many more.