CRASHR PROTOCOL SPECIFICATION

Reza Jhay Lacanlale, Clark Alesna

On behalf of SAIB Inc.

ABSTRACT. The Crashr protocol is a decentralized digital asset marketplace that introduces multi-asset trading capabilities on the Cardano blockchain. By expanding beyond the traditional ADA-NFT exchange model, Crashr enables users to trade a wide array of digital assets, including fungible and non-fungible tokens, in a single transaction. This protocol uses the JPG Store[1]'s robust v3 contract as its foundation. This document outlines the key differences between the JPG Store[1] v3 protocol and the Crashr protocol, and the offchain architecture that supports the Crashr platform.

1. Introduction

The Crashr protocol is a decentralized digital asset marketplace on Cardano network.

The protocol is built upon the open-source JPG Store[1] v3 contract developed in Aiken. The JPG Store[1] v3 contract is a robust smart contract that facilitates secure and straightforward asset exchange. It is designed to handle NFT-ADA transactions optimized for processing bulk orders.

This document will outline the functionalities from the JPG Store[1] v3 contract that are retained in the Crashr protocol, as well as the new features introduced by Crashr. The Crashr protocol aims to enhance the trading experience on the Cardano blockchain by enabling arbitrary asset trading, including ADA, NFTs, and fungible tokens, in a single transaction.

This document will also explain the high level offchain architecture supporting the Crashr platform, as well as the use-cases of these components for both users and third party applications.

2. Crashr Protocol Overview

The protocol is designed to use a single smart contract that can handle all the operations needed by the Crashr platform. This includes Listing, Offer, Buy/AcceptOffer, and Cancel/Update.

- A listing operation is a simple transaction that sends the assets to the marketplace contract and specifies the assets the user wants in return in the datum.
- An offer operation is not exactly different from a listing operation, in fact they are the same. The difference is that an offchain component is needed to identify an offer from a listing by checking if there is an existing listing currently for the assets the user wants to get in return.
- A buy operation is a transaction that attempts to unlock the assets locked by the owner of the listing. The buyer must satisfy the payouts set by the owner as well as the royalty and marketplace fees.

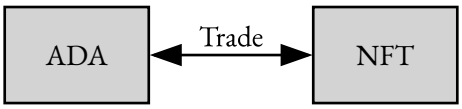
Note:
Payout is a data structure in the smart contract that holds the address of the recipient and the assets they will receive. The datum holds a list of payouts which typically includes the seller payout and royalty payout(s). Marketplace payout is enforced by the contract that is why it's not included in the list of payouts.

- An accept offer operation is also the same as a buy operation, like the offer transaction, to identify an accept offer transaction, an offchain component is needed to check if the user is accepting an offer. In the onchain layer, there are no differences.
- A cancel operation is a transaction that allows the owner to unlock the assets they locked in the marketplace contract. The only requirement of this operation is the owner's signature.
- An update operation is similar to a cancel operation but that owner can unlock the assets and lock new assets or update the datum of the listing then send them back to the marketplace address.

Note:
A user can satisfy multitple listing/offers in a single transaction. The only requirement is that the user must satisfy all the payouts, and each listing/offer must have a separate marketplace fee output, and it's assumed that for each listing/offer, the marketplace fee output is always the first.

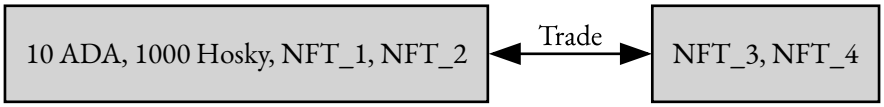
2.1. Traditional Limitations

In Cardano, the current digital asset marketplaces typically operate within a straightforward framework, where users can trade ADA for NFTs or vice versa. While this model has proven effective in normal use-cases, we believe that it falls short in accommodating the diverse and complex trading needs of users in the digital asset space.



2.2. What Crashr Protocol Offers

The Crashr protocol introduces a new feature that allows users to trade any arbitrary combination of assets in a single transaction. This includes trading ADA for NFTs, NFTs for ADA, NFTs for NFTs, ADA and NFTs for other NFTs, and even complex trades involving multiple fungible tokens and NFTs. This flexibility opens up a wide range of trading possibilities, enabling users to engage in more dynamic and diverse transactions on the Cardano blockchain.



3. The Marketplace Smart Contract

The Crashr protocol retains the same functionality as the JPG Store[1] v3 contract with a few modifications in the validation logic to support multi-asset trades. This section is devoted to explaining the core functionalities of the original contract and the enhancements made by Crashr.

3.1. Listings and Offers

One of the key differences between JPG Store[1] v3 and Crashr is that the JPG Store[1] uses different contract or validator for listing and offer transaction. In Crashr, we use the same contract for both listing and offer transactions. The nuance is that there needs to be an offchain component to identify if the user is making an offer or a listing because on the onchain layer there is no difference between the two.

This change simplifies how the users interact with the marketplace contract.

3.2. Payouts

We've updated the shape of the payouts to accommodate multi-asset trading. The payouts are a list of addresses and assets that the user and royalties 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 the transaction.

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 understand payouts better, let's consider an example where a user wants to trade 10 ADA and NFT_1 for NFT_2. In this example trade, the user will send the assets to the marketplace contract with the following datum:

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

In this example, the user only wants to receive NFT 2 in return for the trade. The `policy_id_of_NFT_2` is the policy ID of the NFT_2 asset, and the `NFT_2` is the asset name. The `owner_pkh` is the public key hash of the user who locked the assets in the marketplace contract.

This trade also do not specify a royalty payout. If there is a royalty payout, it will be included in the payouts list with the address of the original creator of the NFT and the amount of ADA they will receive.

3.2.4. Collection Offer Payout Example

In some cases, the user may want to receive any asset from a particular collection. This is also called a collection offer. This type of payout is also supported by the Crashr protocol. In this case, the user only needs to provide the policy ID of the collection and the number of assets they want to receive. The buyer can fulfill this request by sending any asset that falls under the given policy ID. The payouts would look like this:

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

3.3. Marketplace Fee

The marketplace fee is a fee enforced by the protocol to support the maintenance and development of the marketplace. The fee is hardcoded in the contract and is always the first output in the transaction.

3.3.1. JPG Store v3 Fee Calculation

In the JPG Store[1] v3 contract, the marketplace fee is calculated as 2% of the total payout amount. Since the protocol only supports NFT-ADA trades in the original contract, the fee calculation is straightforward.

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

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

In the Crashr protocol, the marketplace fee calculation is adjusted to account for the diverse trading possibilities now that it supports multi-asset arbitrary trades. The fee for ADA payouts remain the same as the original contract.

3.3.3. Unique Token Fee

Since the protocol now supports trades involving multiple assets we needed to account for the tokens included in the payout. To accommodate this, we introduced a unique token fee. Unique token fee is a fee that must be paid for each unique asset requested by the owner of the listing. The fee is also hardcoded in the contract and is currently set to 1 ADA per unique asset. This fee is enforced by the contract. For fungible token transactions, the unique token fee is also applied if the quantity of the tokens is below 100. But for quantities 100 and above, the fee is the same as the original marketplace fee which is 2% of the total payout amount for that token.

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

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 feature in the NFT space that allows creators to earn a percentage of the sale price whenever their NFT is traded. In the JPG Store[1] v3 contract, royalty calculations are straightforward as the protocol only supports NFT-ADA trades. The royalty fee is calculated as a percentage of the total amount of the listing.

The Crashr protocol also supports royalties and allows the user to specify a royalty fee for the original creator of the NFT. This fee is calculated offchain and is expected to be in ADA, based on the estimated price of the assets at the time of listing. The royalty fee is not enforced by the contract but can be added to the payouts list. The minimum requirement for a royalty fee is 1 ADA.

Note:
Due to the unique nature of multi-asset trades, the royalty fee calculation depends heavily on the assets involved in the trade. If there are multiple assets from different projects, each project's royalty address may be included in the payouts list, provided the ADA value meets the minimum requirement of 1 ADA.

3.5. Treasury

The treasury wallet will hold the marketplace fees collected from successful trades. This wallet is managed by the Crashr team and is used to fund the development and maintenance of the marketplace. Aside from the development and maintenance, a percentage of the treasury funds may be allocated to community initiatives, partnerships, and other strategic activities that benefit the Crashr ecosystem.

TODO: Fee breakdown, treasury address?,

3.6. Security

We understand that security is paramount in a decentralized marketplace. The Crashr protocol maintains the security features of the JPG Store[1] v3 contract, ensuring that all assets are secure and that transactions are executed with precision and accuracy.

Since the contract have been modified to support multi-asset trades, we have taken additional measures to have the changes audited by security experts to ensure that the protocol remains robust and secure. As of the time of writing, this contract is still under audit.

4. Marketplace Transaction Flow

To better understand the operations supported by the Crashr protocol, we will delve into the transactional flow for each key action: Listing, Buying, and Canceling/Updating a listing. This section will also visualize how the actual transaction will look like on the Cardano network. These transactions are the core operations that drive the marketplace's functionality and enable users to engage in a wide range of asset trades.

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**.

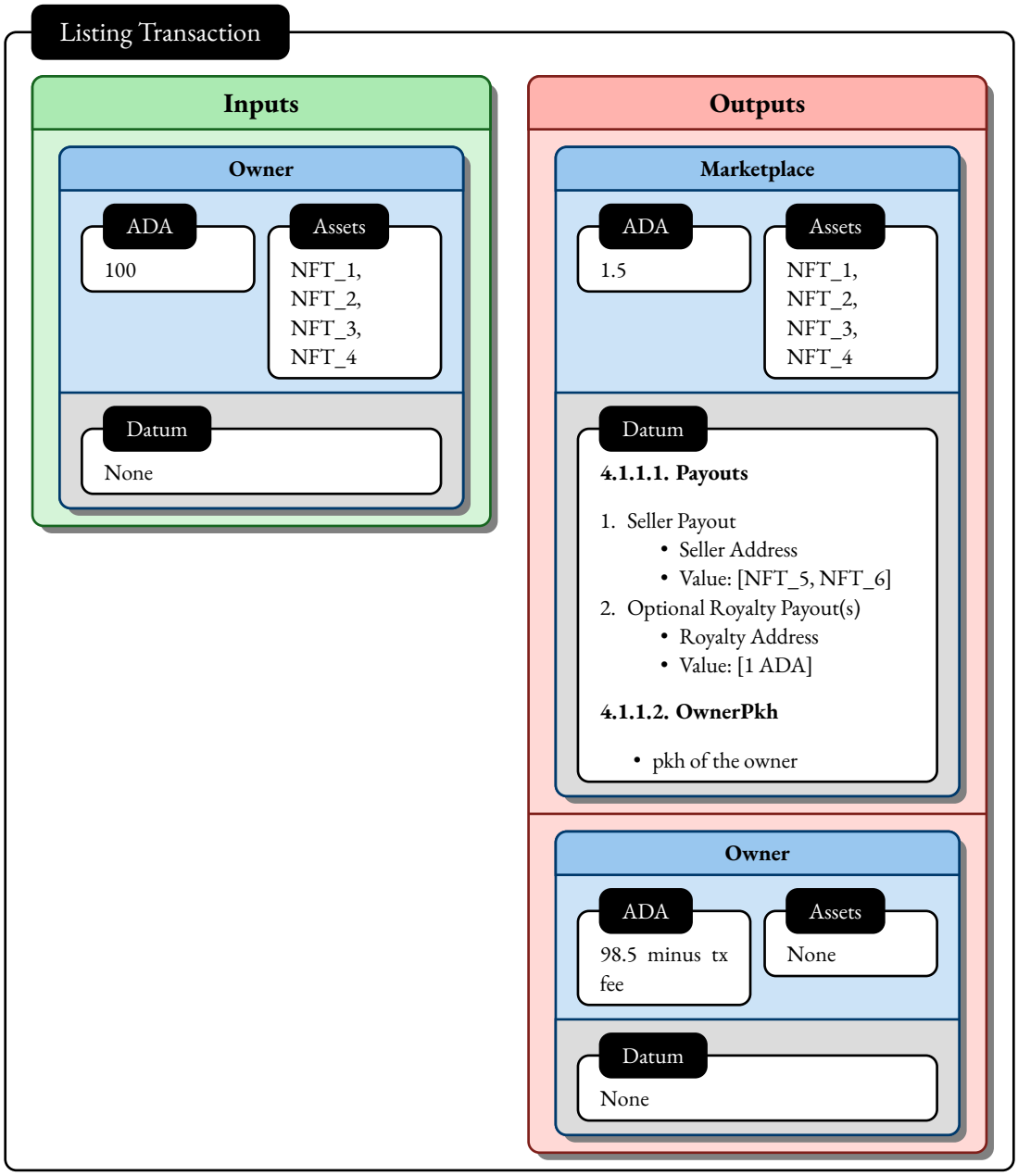


Figure 1: Sample *listing transaction* where the owner locks 4 NFTs and 1.5 ADA in the marketplace validator address with a datum specifying the seller's address and the assets they want to receive in exchange and optional royalty payouts.

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.

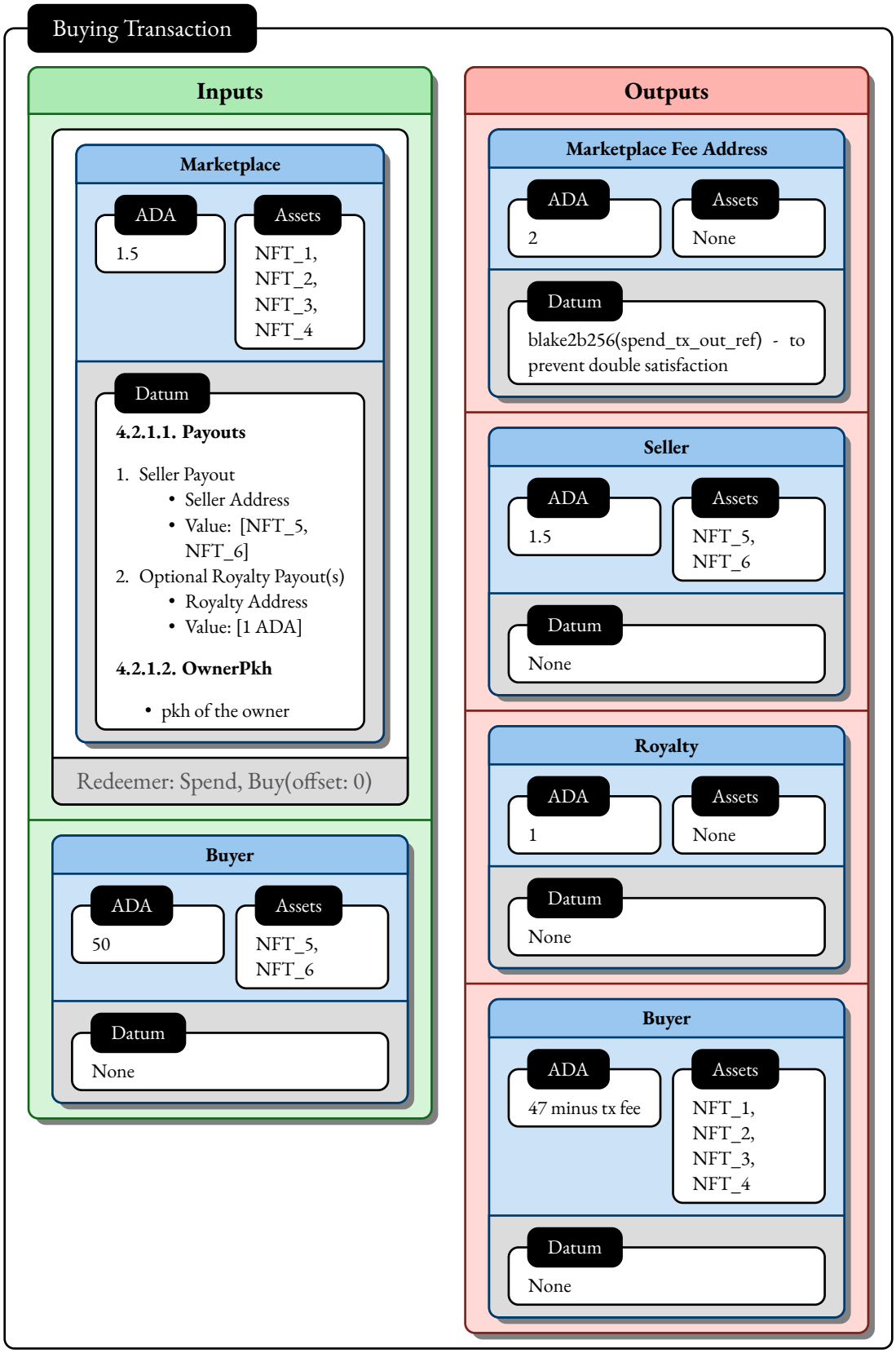


Figure 2: Sample *buying transaction* where the buyer satisfies the seller's requirements by sending the requested assets to the sender's address, the buyer must also satisfy royalty payout requirements if applicable and send the 2% fee + 1 ADA for each unique asset requested. Minimum marketplace fee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

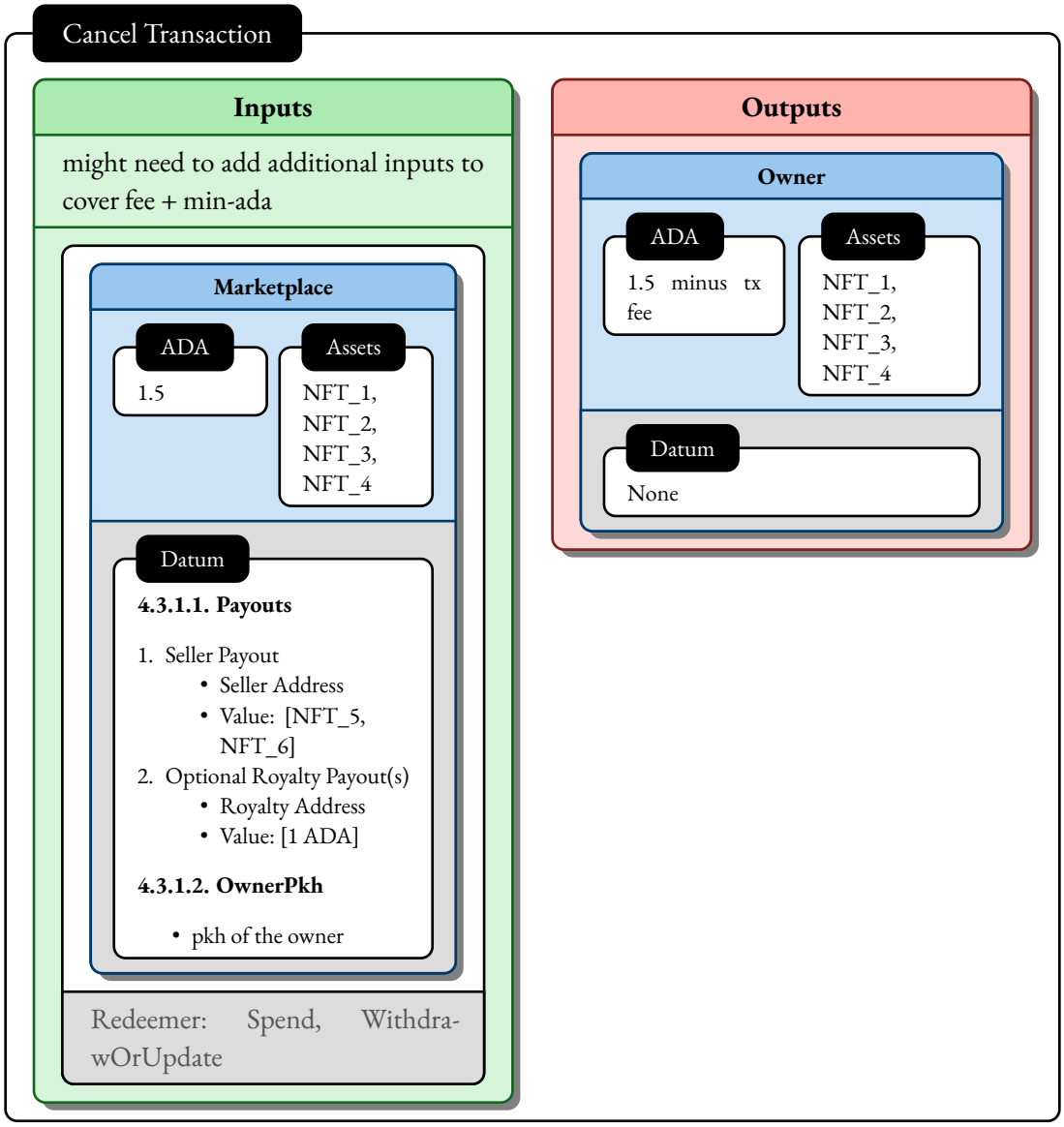


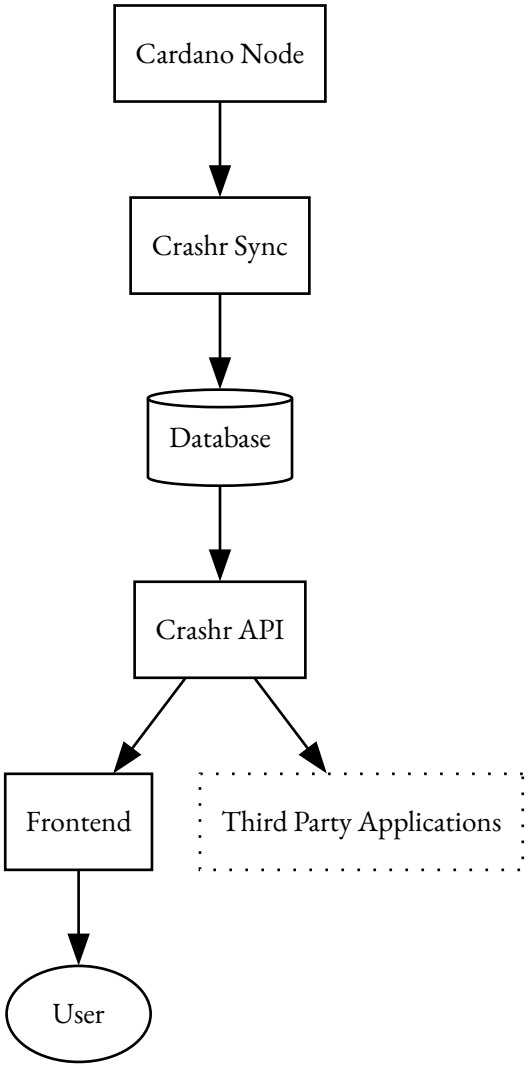
Figure 3: Sample *cancel transaction* where the owner cancels the listing and retrieves the locked assets.

Note:
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.

5. Offchain Architecture

The offchain architecture of the Crashr platform is designed to support the core functionalities of the marketplace, ensuring seamless and efficient trading operations. The offchain components play a crucial role in enhancing the user experience, providing real-time data processing, and enabling complex multi-asset trades. All offchain components are written in C# using the .NET Framework.

5.1. High Level Overview of Offchain Components



5.2. Crashr Sync

Crashr Sync is a critical component of the offchain architecture responsible for processing data from the Cardano blockchain in real-time and saves the processed data, block-by-block, to the database. This component ensures that the marketplace remains up-to-date with the latest blockchain activities, providing users with accurate and timely information for their trading activities. Crashr Sync is built on the foundational technologies of the Cardano Sync by SAIB Inc. and the Pallas library by TxPipe[2], enabling efficient data processing and indexing of blockchain data.

This component is inspired by TxPipe[2]'s Scrolls project, which provides a robust framework for indexing blockchain data for more efficient querying and data management. Crashr Sync leverages a series of specialized reducers to index specific types of data relevant to the marketplace's operations, including smart contract activities, asset metadata, and token prices. These reducers ensure that the marketplace remains dynamic and responsive.

Note:
Crashr Sync relies on the Cardano Node to access 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 crucial 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

TODO: what features do we know we want to add? what features are we considering? what are the next steps for the project?

7. Acknowledgements

We are very proud of the work we have put into this protocol but we could not have done it without the great open-source contributions from the Cardano community. We would like to thank the following projects for making this possible:

- **JPG Store**: For the foundational contract that Crashr is built upon.
- **TxPipe**: For providing the open-source tools that make Cardano development so much more developer-friendly.
- **CardanoSharp** & **Orion**: For their open-source Cardano Cryptographic and Serialization library for .NET applications.
- **TBD**: For our audit.