Efficient Runes Marketplace Specification

Purpose

This specification describes a protocol for an improved Runes marketplace where sell orders can be filled partially using the `OP_CAT` opcode. This improves the UX and reduces transaction fees.

Background

The Runes protocol is one of the 2 most popular fungible token standards on Bitcoin. It allows for creating, minting and transferring tokens using `OP_RETURN` transactions.

Runes transactions are just arbitrary data encoded in `OP_RETURN` for the miners so they cannot be miner validated. In order to enforce validity, all current Runes marketplaces must rely on partially-signed Bitcoin transactions (PSBTs). The seller signs a transaction using `SIGHASH_SINGLE | ANYONECANPAY` that has their Runes UTXO as an input and an output to them with a specific amount. Anyone who wants to buy the tokens can add their inputs and outputs to make the transaction valid and broadcast it to the network.

Using PSBTs allows for a buyer to purchase all tokens in a sell offer on none at all. This leads to bad UX because the sellers are incentivized to split their tokens into multiple UTXOs in order to increase the chance of their orders being completed.

Example of current Runes marketplace UX

The seller (**S**) has 100 R tokens and wants to sell 50 of them
The buyer (**B**) has Bitcoin and wants to buy 40 R tokens

1. **S** has to split their Runes UTXO from 1x100 => 2x50.
2. **S** creates PSBT that signs for RUNE_INPUT committing to an output BTC_OUTPUT paying themselves the sale price and sends it to a Runes marketplace.
3. **B** opens a Runes marketplace and checks all listings for buying R token, sorted by price per token
4. **B** sees the **S**'s listing but they can't purchase it because it is for 50 tokens and **B** wants to buy 40. **B** has to search for another listing or wait for **S** to split their UTXO into 5x10 and buy 4 of them.

OP_CAT

If `OP_CAT` is activated on Bitcoin, it would enable the creation of covenant transactions - transactions that impose specific spending conditions. This functionality could facilitate the

construction of smart Runes token sell offers, allowing partial fulfillment of orders while returning the remaining tokens to the same covenant. Such a feature would enhance the user experience for sellers, who would no longer need to split their tokens into multiple UTXOs, and for buyers, who could purchase portions of sell orders.

Requirements for a smart Runes marketplace

We believe that a good Runes marketplace should have the following properties:

1. Anyone should be able to buy and sell tokens.
2. Users should be able to buy only part of the tokens in a sell order.
3. Users must be forced to buy the full amount of tokens, if the remaining tokens in the offer are below a threshold.
4. The seller should be able to cancel their sell order.
5. The seller should be able to change the exchange rate of their sell order.

Specification

Bitcoin Transactions

Create Sell Order Transaction

INDEX	INPUT	OUTPUT
1	**(100 Runes tokens)** 546 satoshis	**(100 Runes tokens)** 546 satoshis
2	*funding input*	OP_RETURN state
3		OP_RETURN Runestone

The “create sell order” transaction sends Runes to a Tapscript covenant address. This transaction is signed by the seller and sent to the server along with its parameters. The covenant enforces that the transaction can be spent in 4 ways: “sell”, “final sell”, “update exchange rate” and “cancel order”.

Covenant Parameters (constant):

- Seller address
- Minimum token threshold
- Token ID

Covenant State (mutable)

- Exchange rate
- Token amount

This transaction is not required to be broadcasted to the blockchain. It can be stored by the marketplace and only be broadcasted alongside a sell transaction. That way if the seller decides to change the exchange rate or cancel the order, it can be done off-chain.

Buy Transaction

INDEX	INPUT	OUTPUT
1	**(100 Runes tokens)** 546 satoshis	**(75 Runes tokens)** 546 satoshis
2	*funding input*	OP_RETURN state
3		OP_RETURN Runestone
4		**(25 Runes tokens)** 546 satoshis
5		*payment output*

Anyone can create a “buy” transaction to buy some of the tokens in the sell order and return the rest to a new covenant address.

The “sell” transaction validates that:

- `vout[0]` has the covenant’s `scriptPubKey`.
- `vout[1]` is `OP_RETURN` state with a modified amount.
- `vout[2]` is correctly formatted `OP_RETURN` Runestone that sends the correct amount of tokens to `vout[3]`.
- pays the correct amount to the seller at `vout[4]` based on the exchange rate.
- the new amount is greater than the threshold.

Final Buy Transaction

INDEX	INPUT	OUTPUT
1	**(25 Runes tokens)** 546 satoshis	OP_RETURN Runestone
2	*funding input*	**(25 Runes tokens)** 546 satoshis
3		*payment output*

Anyone can create a “final buy” transaction to buy all remaining tokens in the sell order.
The “final sell” transaction validates that

- the `vout[0]` is correctly formatted `OP_RETURN` Runestone that sends the full amount of tokens to `vout[1]`.
- pays the correct amount to the seller at `vout[3]` based on the exchange rate.

Update Exchange Rate and Cancel Order Transaction

Updating the exchange rate

INDEX	INPUT	OUTPUT
1	**(100 Runes tokens)** 546 satoshis	**(100 Runes tokens)** 546 satoshis
2	*funding input*	OP_RETURN state
3		OP_RETURN Runestone

Canceling the sell order

INDEX	INPUT	OUTPUT
1	**(100 Runes tokens)** 546 satoshis	**(100 Runes tokens)** 546 satoshis
2	*funding input*	OP_RETURN Runestone

The owner of the listing (the seller) has the ability to spend from the covenant without restrictions. They can withdraw part of the tokens, update the exchange rate and close the offer.

Implementation

The source code is hosted on GitHub. It contains 3 folders:

- contracts - Bitcoin scripts written is Scrypt
- backend - a simple Express server with SQLite database that communicates with an Ord indexer instance and stores the listings
- frontend - a Next.js web application that allows users to buy and sell Runes

Findings

Non-standard transactions

The marketplace transactions have 2 `OP_RETURN` outputs: one for the covenant state, caboose, and one for the Runes transfer. This makes them non-standard and most nodes won't propagate them. This limitation can be solved in the future with TXID reflection.

Transaction size

Transaction size is ~5kB but because most of the data is in the witness, its virtual size is ~1.5kB. At the moment the whole script is in one Tapleaf, moving the 3 spending conditions: buy, full buy and updateOrCancel to different Tapleafs would probably reduce the size to ~800B.

Multiplication

The covenant uses multiplication for calculating the bitcoin amount that needs to be sent to the seller using the equation: `btcAmount = purchasedTokens * exchangeRate`.

Bitcoin doesn't have a multiplication opcode so it needs to be simulated using addition. The current implementation uses u15 multiplication from the sCrypt repository. This means the maximum tokens that can be purchased from the covenant and the maximum exchange rate of satoshis for 1 token is 32767 so the marketplace is not suitable for Runes with very large supply or big divisibility

Maximum purchasable tokens based on the token divisibility:

Token Divisibility	Maximum purchasable tokens
0	32767
1	3276,7
2	327,67
3	32,767

Arithmetic inputs in Bitcoin are limited to signed 32-bit integers, which makes the max amount that can be stored in unsigned integers ~2.1B. By using u31 multiplication we can increase the maximum purchasable tokens more than 60,000 times.

Change output

The covenant does not support sending the remaining satoshis to a change output so the buyers must first send their bitcoin to a new output with amount: `btcPaymentAmount + (~1500 sats * currentSatsPerVByteFee)`, otherwise they risk paying high fees. The covenant can easily be modified to support additional outputs in the future.