ZELDHASH PROTOCOL

Hunt for Bitcoin's Rarest Transactions and Earn ZELD.

By Ouziel Slama

1 Motivations

- For the thrill of the hunt. Every transaction becomes an opportunity to discover something rare — a digital treasure hidden in plain sight on the blockchain.
- These patterns of leading zeros aren't just rare — they could also enhance compression, potentially streamlining blockchain storage and processing efficiency.
- Anyone can earn ZELD by hunting rare transactions — no single-winner-per-block like in Bitcoin block mining. The hunt is open to all.
- If successful, ZELD tokens could eventually reimburse transaction fees — rewarding hunters who uncover the rarest finds!

2 ZELD mining

To mine ZELD you must broadcast a Bitcoin transaction whose txid starts with at least 6 zeros. The reward is calculated based on how your transaction compares to the transaction(s) with the most leading zeros in that block:

- In a given block, transactions starting with the most zeros earn 4096 ZELD
- Transactions with one fewer zero than the best earn 4096 / 16 = 256 ZELD
- Transactions with two fewer zeros earn 4096 / 16 / 16 = 16 ZELD
- etc.

The formula used is therefore as follows:

```
reward = 4096 / 16 ^ (max_zero_count - zero_count)
```

Where `max_zero_count` is the leading zero count of the best transaction in the block, and `zero_count` is the leading zero count of the transaction being evaluated.

Note: Coinbase transactions are not eligible for ZELD rewards.

3 ZELD distribution

Throughout this document, **first non-OP_RETURN output** refers to the lowest-index output in the transaction that is not an OP_RETURN.

ZELD rewards earned by mining always attach to the first non-OP_RETURN output.

4 Moving ZELD

4.1 Method 1: Automatic Distribution

When spending UTXOs that hold ZELD, all ZELD is transferred to the first non-OP_RETURN output by default.

4.2 Method 2: Custom Distribution via OP_RETURN

You can specify exactly how ZELDs should be distributed by including an OP_RETURN output in your transaction with custom distribution data. This allows for precise control over ZELD transfers.

4.2.1 OP_RETURN Format:

- The OP_RETURN script must contain data that starts with the 4-byte prefix "ZELD"
- Following the prefix, the data must be encoded in CBOR format
- The CBOR data must be an array of unsigned 64-bit integers
- Each integer specifies how many ZELDs to send to the corresponding output

4.2.2 Distribution Rules:

- The number of values in the distribution array is automatically adjusted to match the number of non-OP_RETURN outputs
- If the array is too long, extra values are removed
- If the array is too short, zeros are appended
- The total sum of the distribution values cannot exceed the total ZELDs being spent
- If the sum is less than the total, the remainder is added to the first non-OP_RETURN output
- If the sum exceeds the total, the custom distribution is ignored and all spent ZELD attaches to the first non-OP_RETURN output
- Newly mined ZELD rewards always attach to the first non-OP_RETURN output, regardless of the custom distribution

4.2.3 Example:

If you have 1000 ZELDs to distribute across 3 outputs and want to send 600 to the first, 300 to the second, and 100 to the third, your OP_RETURN would contain "ZELD" followed by the CBOR encoding of [600, 300, 100].

Notes:
- If no valid OP_RETURN distribution is found, the automatic distribution (Method 1) applies.
- If all outputs are OP_RETURN, any ZELD attached to the transaction's inputs **and any newly earned reward** is permanently burned because there are no spendable outputs to receive them.
- When multiple OP_RETURN outputs are present, only the last one carrying a valid `ZELD`+CBOR payload is considered.