BIP — Partial Non-Fungibility (PNF) — UID & Transfer-as-Mint Proposal

Title: Partial Non-Fungibility (PNF): UID & Transfer-as-Mint lineage convention for non-fungible Bitcoin outputs
Author(s): Aaron Tolentino (draft), contributors welcome
Status: Draft 1.0 — Request for Discussion
Type: Standards Track / Policy & Best Practice

Created: 2025-09-23

Abstract

This BIP proposes **Partial Non-Fungibility (PNF)**, a convention to enable unique, non-fungible outputs on Bitcoin **without indexing all satoshis** or embedding large data blobs into witness space.

The core of PNF is the **Unique Identifier (UID)** — a compact 32-byte commitment bound to a specific output.
To support transfer and property tracking, this BIP introduces the **Transfer-as-Mint lineage convention**: every transfer of a PNF-marked UTXO must include a new UID that explicitly references the previous UID.

This creates a verifiable lineage chain **(UID → UID → UID)** with minimal on-chain footprint.

Motivation

There is undeniable market demand for **uniqueness** (collectibles, certificates, provenance).

Current approaches (Ordinals, Inscriptions) rely on full satoshi indexing and heavy witness data storage, bloating Bitcoin blocks.

PNF satisfies the same demand with a lighter design:

- UID commitments (32 bytes) instead of large witness inscriptions.
- Lineage chains for transfers instead of global satoshi indexing.
- Minimal reliance on off-chain infrastructure (optional registries for metadata).

This reduces economic incentive for arbitrary inscriptions and provides a **market-compatible alternative**.

Specification

UID Definition

Each PNF output is assigned a UID:

```
UID = SHA256("PNF_v1" || txid || ":" || vout || ":" ||
nonce || ":" || policy_id)
```

- **txid** = transaction ID (hex)
- **vout** = output index (decimal)
- **nonce** = random or user-defined entropy (32–64 bits recommended)
- **policy_id** = short ASCII token defining the namespace (optional)
- **v1** = version marker

The UID is published in an `OP_RETURN` anchor:

```
OP_RETURN [0x01][len(policy_id)][policy_id][32-byte UID]
```

Transfer-as-Mint (Lineage)

When a PNF output is transferred, the new transaction must include a new UID that references the previous one:

```
newUID = SHA256("PNF_v1_next" || prevUID || new_txid || ":"
|| new_vout || ":" || nonce || ":" || policy_id)
```

- **prevUID** = UID of the transferred output
- **new_txid/new_vout** = location of the new output
- **nonce/policy_id** = same rules as mint

The transaction must include an `OP_RETURN` with the new UID:

```
OP_RETURN [0x02][len(policy_id)][policy_id][32-byte newUID]
[32-byte prevUID]
```

This ensures lineage is explicitly visible on-chain.
Indexers and marketplaces can resolve current ownership by following the latest UID in the chain.

Metadata Anchoring

Metadata describing the asset is stored off-chain (IPFS / Arweave / registries).
It must include `fileHash`, `UID`, and canonical serialization.
Optionally, commit-reveal schemes allow confidentiality until transfer.

Freeze Script Template (Optional)

For stronger guarantees, outputs may use a freeze script to enforce reveal conditions:

```
OP_SHA256 <fileHash> OP_EQUALVERIFY <pubkey> OP_CHECKSIG
```

Spending requires both a valid preimage (metadata hash) and a signature from the controlling key.
This ensures controlled reveal and transfer mechanics.

Relay and Policy Recommendations

- **Wallets:** support UID creation and lineage during transfers.
- **Nodes:** may deprioritize transactions with large witness blobs lacking UID anchors.
- **Marketplaces:** should prefer lineage-compliant UIDs and show ownership via the latest UID in the chain.
- **Miners:** may advertise preference for compact UID anchors as efficient blockspace use.

Rationale

- UID anchors avoid bloating witness space with arbitrary data.
- Transfer-as-Mint ensures clear, verifiable ownership history without satoshi indexing.
- Market adoption provides economic incentive to follow this standard over raw inscriptions.

Backwards Compatibility

Fully backward compatible with Bitcoin — only standard scripts (P2WPKH, P2WSH, OP_RETURN) are used.
No changes to consensus rules are required.

Security Considerations

- **Collisions:** prevented by including txid, vout, nonce, and policy.
- **Off-chain metadata loss:** mitigated by redundant anchors (IPFS + Arweave).
- **Privacy:** commit-reveal schemes allow delayed disclosure of preimages.
- **Censorship:** relay/mempool policies are opt-in, not consensus-enforced.

Test Vector (illustrative)

Mint transaction `txA`, `vout=0`

```
UID_A = SHA256("PNF_v1"||txidA||":"||0||":"||
42||":"||"ART")
Anchor: OP_RETURN 0x01 0x03 "ART" UID_A
```

Transfer to new owner `txB`, `vout=1`

```
UID_B = SHA256("PNF_v1_next"||UID_A||txidB||":"||1||":"||
7||":"||"ART")
Anchor: OP_RETURN 0x02 0x03 "ART" UID_B UID_A
```

Ownership chain: `UID_A → UID_B`
Current owner = receiver of `txB:vout=1`

Economic Impact

- UID anchors ≈ 40–70 bytes per transaction (vs kilobytes for inscriptions).
- Encourages efficient blockspace usage.
- Marketplaces benefit from simpler indexing (lineage chain).
- Reduces network congestion and negative externalities from raw inscriptions.

Reference Implementation

Reference SDKs and CLI tools are expected to:

- Compute UIDs.
- Build `OP_RETURN` anchors.
- Handle lineage updates on transfer.
- Integrate with IPFS / Arweave for metadata.

Acknowledgements

Thanks to community participants for early discussions on ordinals, inscriptions, and fungibility. This BIP proposes a compromise that respects Bitcoin's monetary base while enabling uniqueness where markets demand it.

Conclusion

Partial Non-Fungibility (PNF) provides a path to uniqueness on Bitcoin that is efficient, verifiable, and market-aligned.

By combining UID commitments with the Transfer-as-Mint lineage convention, we can satisfy demand for non-fungibility without bloated witness data or global satoshi indexing.

This proposal is open for discussion, refinement, and community adoption.