KILN Teleburn Protocol

Solana → Bitcoin Ordinals Migration Standard

Version	1.0
Status	Final Specification
Date	December 5, 2025
Author	Fevra (KILN)
Repository	github.com/fevra-dev/kiln

1. Abstract

This specification defines the KILN Teleburn Protocol, a minimal standard for verifiably linking the destruction of a Solana NFT to a corresponding Bitcoin Ordinal inscription. The protocol uses a generic `teleburn:` memo prefix designed for ecosystem-wide adoption.

The specification was developed through collaborative review with multiple AI systems (Claude, GPT-4, Grok) to ensure technical rigor, security, and practical adoptability.

2. Background

2.1 What is Teleburn?

Teleburn is a cross-chain migration mechanism where an asset is permanently destroyed on one blockchain while a corresponding asset exists on another. The destruction includes on-chain proof linking to the destination asset, enabling anyone to independently verify the migration.

2.2 Why Generic Prefix?

The `teleburn:` prefix was chosen over implementation-specific prefixes (e.g., `kiln:`) to maximize ecosystem adoption. Any project can implement this standard without branding concerns. KILN is the reference implementation, but the memo format is chain-agnostic and tool-agnostic.

2.3 Why Solana is Different

Solana's architecture differs fundamentally from Ethereum:

1. **Native burns:** Metaplex/SPL Token programs destroy tokens directly (supply → 0)
2. **No black holes:** Unlike Ethereum, no derived address is needed — the token ceases to exist
3. **Memo linkage:** The inscription ID is recorded in a memo attached to the burn transaction

3. Specification

3.1 Memo Format (Normative)

A valid teleburn memo MUST conform to the following grammar:

```
teleburn-memo  ::= prefix ":" inscription-id
prefix         ::= "teleburn"
inscription-id ::= txid "i" index
txid           ::= [a-f0-9]{64}
index          ::= [0-9]+
```

3.1.1 Canonical Example

```
teleburn:6fb976ab49dcec017f1e201e84395983204ae1a7c2abf7ced0a85d692e442799i0
```

3.1.2 Encoding Rules

- ASCII only
- Lowercase hex for txid (MUST NOT use uppercase)
- No whitespace
- No leading zeros in index (except for index "0")
- Total memo length MUST NOT exceed 100 bytes

3.2 Transaction Requirements (Normative)

A valid teleburn transaction MUST:

1. Execute a Metaplex BurnV1 instruction that reduces token supply to exactly 0
2. Include exactly one memo instruction containing a valid teleburn memo
3. Execute both instructions in the same transaction (atomic)

3.3 Versioning

Future protocol versions MUST use a different prefix. The progression is: `teleburn:` (v1) → `teleburn2:` (v2) → `teleburn3:` (v3), etc.

4. Verification Procedure

To verify a teleburn, a verifier SHOULD:

Step 1: Validate Burn Transaction

- Confirm transaction contains a Metaplex BurnV1 instruction
- Confirm token mint supply is now 0

Step 2: Parse Memo

- Extract memo string from transaction
- Validate memo matches grammar (Section 3.1)
- Extract inscription ID

Step 3: Validate Inscription

- Query Bitcoin/Ordinals indexer for inscription ID
- Confirm inscription exists

Step 4: Optional Content Validation

If content equivalence verification is required, fetch both the original Solana NFT metadata (pre-burn) and inscription content, compute SHA-256 hashes, and compare. This step is OPTIONAL and outside the core protocol.

5. Reference Implementation

5.1 Memo Construction

```
function buildTeleburnMemo(inscriptionId: string): string {
  const regex = /^[a-f0-9]{64}i[0-9]+$/;
  if (!regex.test(inscriptionId)) {
    throw new Error('Invalid inscription ID');
  }
  return `teleburn:${inscriptionId}`;
}
```

5.2 Memo Parsing

```
function parseTeleburnMemo(memo: string): string {
  const prefix = 'teleburn:';
  if (!memo.startsWith(prefix)) {
    throw new Error('Not a teleburn memo');
  }
  const inscription = memo.slice(prefix.length);
  const regex = /^[a-f0-9]{64}i[0-9]+$/;
  if (!regex.test(inscription)) {
    throw new Error('Invalid inscription ID');
  }
  return inscription;
}
```

6. Security Considerations

6.1 Data Authenticity

The memo asserts only the linkage between a burned mint and an inscription ID. It does NOT assert content equivalence. Verifiers requiring content matching MUST perform off-chain hash comparison.

6.2 Replay Attacks

Impossible. An NFT can only be burned once. The burn transaction itself is the non-replayable proof.

6.3 False Claim Attacks

A malicious actor could burn an unrelated NFT and claim any inscription ID in the memo. This is a known limitation of the minimal protocol.

Mitigations:

- Application-layer content hash verification
- Marketplace/aggregator provenance checks
- Bidirectional verification via inscription metadata (see companion spec)

7. Design Rationale

7.1 Why teleburn: Instead of kiln:?

A generic prefix maximizes ecosystem adoption. Magic Eden, Tensor, and other platforms can implement this standard without embedding another project's branding. KILN remains the reference implementation, but the format belongs to the ecosystem.

7.2 Why No JSON?

- Larger on-chain footprint (~250 bytes vs ~78 bytes)
- Parsing complexity
- No additional verification value

7.3 Why No Derived Address?

Solana burns destroy tokens via program instruction, not by sending to unspendable addresses. A derived address adds complexity without verification value. The inscription ID is the universal cross-chain identifier.

8. Format Comparison

Aspect	v1.0 (This Spec)	v0.1.x (JSON)	Ethereum
Memo Size	~78 bytes	~250+ bytes	N/A
Prefix	teleburn:	JSON	(derived addr)
Indexing	LIKE 'teleburn:%'	JSON query	Balance check

9. References

- KILN Repository: github.com/fevra-dev/kiln
- Ordinals Teleburning: docs.ordinals.com/guides/teleburning.html
- Solana Memo Program: spl.solana.com/memo
- Metaplex Token Metadata: developers.metaplex.com/token-metadata

— End of Specification —