

CRC NFT

PROTOCOL WHITEPAPER

NFT state. Bitcoin settlement.

A technical whitepaper for unique assets on Bitcoin.

CRC-BASED NFT PROTOCOL / VERSION 1.0

100

Entries per asset event

64 bytes

Issuer signature

3

Service confirmations

Collection identity. Signed events. Deterministic state.
A verifiable asset history carried by Bitcoin transactions.

ABSTRACT / SCOPE

A protocol for NFTs. A record on Bitcoin.

CRC NFT Protocol. Version 1.0.

CRC NFT defines a CRC-based application protocol for individually identified assets on Bitcoin. Its collection identifiers, event grammar and deterministic registry specify how assets are issued, transferred and permanently retired. A rules hash identifies the exact reference implementation used to interpret that history.

An issuer Seal binds each event to its transaction arrangement. Bitcoin validates BTC spending and orders transactions in blocks. Protocol verifiers authenticate the Seal and reconstruct NFT state. A reference client may wait for three block confirmations before presenting a valid operation as complete.

This whitepaper specifies collection identity, event encoding, authorization, state transitions and validation. Wallet interaction and confirmation handling are integration profiles. Collection artwork, campaigns, prices and commercial promises are outside the protocol specification.

Three layers. Defined responsibilities.

The chain records transactions. The registry interprets NFT events. The issuer checks authorization intent before signing. Verifiers reconstruct the registry independently while retaining the issuer's explicit authorization boundary.

In this whitepaper

Architecture and trust	03	Parallelism and confirmations	11
Collection identity	04	Funding and relay constraints	12
Canonical event encoding	05	Wallet authorization	13
Event definitions	06	Collection lifecycle	14
Seal authorization	07	Security model	15
Asset state and checkpoints	08	Reference parameters	16
Validation and indexing	09	Terminology	17
Event and funding binding	10	References and scope	18

Protocol architecture.

Authorization, funding and asset interpretation remain separate.

Participant	Responsibility
Holder wallet	Proves address control and authorizes its own BTC funding inputs.
Issuer / signing service	Checks authorization intent, then signs the event bound to its transaction arrangement.
Bitcoin network	Verifies BTC spending conditions and provides canonical transaction ordering.
Protocol verifier / indexer	Validates the issuer Seal and events; reconstructs asset state from the effective chain.
Content service	Provides catalogs, metadata and files for comparison with committed hashes.
Application client	Presents asset state, prepares wallet interactions and follows transaction status.

One transaction can carry two results.

A transaction may record a valid NFT event and carry BTC outputs at the same time. Funds are governed by Bitcoin spending rules; NFT meaning is governed by the selected CRC NFT rules. A confirmed transaction still needs valid event interpretation before a client can treat its asset transition as accepted.

Authorization has an explicit boundary.

The issuer is the protocol authorization authority. The event validator checks the issuer signature; it does not independently demand the current NFT owner's message signature. Holder consent is checked by the signing service, so issuer integrity remains a trust dependency.

Replay produces the asset record.

Verifiers replay the same effective chain and rules to obtain matching collection state. Content hashes enable file checks. A user interface presents that validated registry; it does not define NFT identity.

Identity begins with deployment.

Network, collection and serial together identify an NFT.

Collection ID = genesisHash : deploymentTxid

NFT ID = collection ID + serial

The genesis hash separates Bitcoin networks; the deployment transaction distinguishes collections on a network. A serial is unique only inside its collection. These fields prevent an identical serial or catalog from being mistaken for another asset.

Deployment field	Purpose
maxSupply	Fixes the collection's serial boundary; the reference codec supports 1 to 4,000.
catalogHash	Commits the catalog that maps serials to asset content.
profileHash	Commits collection-specific descriptive data.
rulesHash	Identifies the exact event and validation rules.
issuerKey	Provides the 32-byte issuer key for Schnorr verification.
treasuryScript	Commits the collection's Taproot receiving script.

DEPLOY carries a zero collection reference. The registry derives the full collection identity from the actual deployment txid afterward; inserting that txid in the event would create a circular dependency. Later events carry the original deployment txid as their collection reference.

A content commitment is a checkable promise.

Catalog and file hashes let a verifier detect a file that differs from the committed content. Files remain off-chain and must be available from a content service. The chain commits their identity; it does not provide the file bytes or ensure continued delivery.

One event. One canonical payload.

The first transaction output carries the event and its Seal.

A protocol transaction has exactly one OP_RETURN output, at index 0, with a zero BTC value. Its script contains one canonical data push. This fixes where the event is located and prevents ambiguous command selection.

Payload field	Bytes	Meaning
Framing marker	5	0x43524e4603, the exact reference framing bytes.
Event opcode	1	Selects the typed operation.
Collection reference	32	Original deployment txid; zero for DEPLOY.
Operation ID	16	Unique authorization identifier within the collection.
Event-specific body	Variable	Fields required by the selected event.
Issuer signature	64	BIP340 Schnorr signature appended to the body.

The common header is 54 bytes. Integers are unsigned and big-endian. Hashes, scripts and public keys are raw bytes, not textual hex or addresses. The framing marker is a fixed codec identifier; the public specification edition is Version 1.0.

Encode one meaning.

The decoder rejects truncation, trailing bytes, invalid fields and a script that does not exactly match canonical re-encoding. This prevents multiple accepted byte forms from representing the same event.

MINT and MOVE accept a 34-byte P2TR recipient script or a 22-byte P2WPKH recipient script. RETIRE uses a zero recipient length. Entries are strictly increasing and distinct, with 1 to 100 serials, so a batch has one unambiguous order and no duplicate asset.

Each serial must also fit the deployed maximum supply. The codec's 4,000-serial ceiling is a reference-protocol bound, not a Bitcoin consensus limit.

Six events.

A complete asset lifecycle.

Each command has a specific effect and validation purpose.

Command	Effect	Required condition
DEPLOY / 0x01	Create a collection.	Zero collection field, supported supply and treasury script, accepted rules and valid issuer signature.
MINT / 0x02	Issue unused serials.	Issuance remains open; every serial is within supply and has never been issued.
MOVE / 0x03	Change the owner script.	Each asset is active; its previous event and next version match current state.
RETIRE / 0x04	End circulation.	Uses the same prior-state checks as MOVE; clears ownership and preserves history.
CLOSE_MINT / 0x05	Close future issuance.	The collection is active and issuance has not already been closed.
UPGRADE / 0x06	Import continuous state.	Binds an existing authority-anchored registry and its prior checkpoint under the original issuer.

Batch operations are all-or-nothing.

All entries must validate before the new collection state is committed. A single invalid serial or stale predecessor rejects the asset event, so an application cannot infer partial delivery from a rejected batch.

All assets in a MINT or MOVE share one recipient script. MOVE and RETIRE each carry the affected asset's prior event txid and next version. Per-asset references make the transition locally verifiable.

Retirement retains identity.

RETIRE leaves a terminal registry record. The serial remains occupied, its history stays available and MINT cannot reuse it. CLOSE_MINT independently closes future issuance while existing active assets retain their movement and retirement rules.

Seal authorization.

The issuer Seal uses BIP340 Schnorr authorization [1].

```
digest = TaggedHash(D,  
genesis || rulesHash || encodedEvent || UTF8(txView))
```

encodedEvent contains the full 54-byte header and all event-specific fields, excluding the signature. D is the immutable signing-domain byte string in the content-addressed reference rules. Verifiers recover its exact bytes from those rules to preserve signature compatibility.

Committed component	Validation purpose
Complete event body	Binds the collection, operation ID, serials, recipient and required predecessor fields.
Network and rules hash	Separates signing across networks, rule sets and signature purposes.
Transaction version and locktime	Fixes the transaction control fields.
All ordered input outputpoints and sequences	Fixes which funding inputs the issuer authorized.
All later output amounts and scripts	Binds BTC payments and change after the event output.

TaggedHash prefixes the message with SHA-256(D) twice. txView uses canonical JSON: sorted keys, preserved array order, exact integers and no whitespace. Later outputs are normalized to value and scriptpubkey.

Avoid a circular signature.

The command-output script, issuer signature, transaction's own txid, scriptSig, witness, input amounts and previous-output scripts are excluded from this digest. The encoded event is committed separately. Wallet signatures and Bitcoin consensus retain their own funding checks.

DEPLOY uses the issuer key in its body; later events use the recorded collection key. The applicable rules hash must match accepted reference rules so a signature cannot select another evaluator.

State follows the asset.

A serial keeps its identity as ownership changes.

01	02	03
UNISSUED	ACTIVE	RETIRED
Transition	State update	
MINT	Create an active asset at version 1; record recipient and event txid.	
MOVE	Replace the owner; record the new txid and increment the version.	
RETIRE	Clear the owner; record the final txid, next version and retired status.	

For MOVE or RETIRE, previousTxid must equal the asset's latest valid event and version must equal its current version plus one. This rejects outdated or competing transitions that reference a state no longer current.

A checkpoint for each collection.

stateHash is SHA-256 of canonical JSON containing the preserved continuity baseline, the deployment digest when present, issuanceClosed, sorted accepted operation IDs and full NFT records sorted by serial.

The collection sequence is excluded from this checkpoint. Normal asset events do not sign a global sequence or the full stateHash. The hash supports deterministic registry comparison while independent assets keep independent authorization dependencies.

An ordinary BTC spend is not an NFT move.

NFT ownership is the receiving script in the validated registry. Spending BTC associated with that address does not itself change asset ownership; a valid MOVE event advances the NFT state.

Reconstruct. Validate. Commit.

A deterministic verifier follows the effective Bitcoin chain.

01 **Read the ordered transaction.**

Locate the unique command at output 0 and validate the supported transaction view.

02 **Decode canonical event bytes.**

Reject malformed fields, trailing data and a failed re-encoding match to preserve one interpretation.

03 **Resolve collection and rules.**

Derive a new deployment identity or locate the active collection; select the accepted rules hash.

04 **Verify issuer authorization.**

Check the Schnorr Seal against the transaction view and the appropriate issuer key.

05 **Reject replay and stale state.**

Check operation uniqueness, supply boundaries, prior txids, versions and issuance status.

06 **Commit the complete transition.**

Apply all entries atomically, record the operation ID and rebuild the collection checkpoint.

Repeated processing of the same txid is idempotent. A reused operation ID within the collection is rejected, because transaction identifiers alone do not prevent authorization replay.

The effective chain controls ordering.

The indexer stores block checkpoints and rolls back when a reorganization replaces history. It replays the replacement chain and rebuilds affected collections. Clients recompute confirmation counts and avoid treating rolled-back events as completed operations.

Malformed or invalid events leave ordinary asset state unchanged. Spending an authority-anchored collection's controlling output without valid continuity can suspend that collection, because its previous authorization chain no longer exists.

Bind the event. Commit the funding.

The Seal commits the arrangement; Bitcoin verifies its spend.

Construct the event.

The issuer prepares a canonical DEPLOY, MINT, MOVE, RETIRE, CLOSE_MINT or continuity event. Asset entries and recipient scripts reflect the intended registry transition. Required prior-state references are resolved before signing so the event advances the current asset state.

Commit the actual outputs.

The transaction view binds every output after the command to its exact satoshi value and script. NFT ownership is recorded by the event's recipient script, while BTC outputs describe spending and change. A separate BTC carrier output for each NFT recipient is not required by the normal event grammar.

The issuer Seal does not require a commercial price or retirement payout. Any BTC transfer included with an event is part of the signed output arrangement, but the registry does not infer a quote or economic promise from that event.

Authorize the funding.

A wallet receives the transaction and its input-signing data in a PSBT [2]. It checks the spending recipients, input amounts, fee and change, then signs its controlled inputs. Bitcoin consensus validates those funding signatures independently of the issuer's event signature.

01

Build event

02

Seal arrangement

03

Sign inputs

04

Broadcast

Preserve the signed arrangement.

Changing a committed input, sequence, output, version or locktime requires a matching new Seal. A transaction with an invalid or mismatched Seal may still be a valid Bitcoin transaction, but cannot create the intended valid NFT state transition.

Independent assets. Independent progress.

Per-asset dependencies allow independent authorization.

MINT for a distinct unused serial does not reference the previous issuance transaction. Normal MINT, MOVE and RETIRE events require no global authority UTXO, collection sequence or full-state signing dependency. Different assets can proceed independently with non-conflicting funding inputs.

A subsequent action on the same asset still references its latest valid event. A verifier applies it only when the required predecessor is current; distinct assets have no shared predecessor. Bitcoin inputs also need to remain available and non-conflicting.

Client confirmation profile.

0**Not yet in a block**

Follow propagation; no effective-chain event has been applied.

1-2**Observed on-chain**

Validate the event and keep its confirmation status visible.

3+**Reference threshold**

A valid operation meets the reference completion profile.

The first containing block counts as one confirmation. Two further valid blocks reach three. This is a reference client completion policy; confirmation count is not included in the event signature or core event-validity checks.

Three confirmations do not mean inclusion within three blocks of starting an operation, and they do not eliminate reorganization risk. Propagation, miner policy and block timing still affect the wait.

Funding remains an independent constraint.

Parallel authorization does not make an already spent coin available again. A client must select valid inputs and distinguish unconfirmed change from confirmed funding. A service can add further scheduling rules, but those rules do not redefine the NFT event grammar.

Funding and relay.

The event grammar and Bitcoin network policy both apply.

Miner fee = total inputs - total outputs

1 BTC = 100,000,000 sats

Input and output amounts determine the miner fee. Virtual size and a chosen sat/vB rate guide construction. Bitcoin Core estimatesmartfee [3] estimates rates; inclusion and timing remain subject to network conditions.

Constraint	Why it matters
Input availability	Each UTXO must be unspent and meet script and maturity conditions.
Output minimums	Script-specific dust policy can prevent an output from being relayed.
Data-carrier policy	Large OP_RETURN payloads may exceed relay policy.
Actual fee and change	Inputs minus outputs fix the actual fee and change.
Signature compatibility	Committed field changes require a matching new Seal.

Protocol capacity is not relay policy.

The codec accepts up to 100 asset entries. MOVE and RETIRE are larger than MINT because each entry includes prior state. Clients measure the transaction and check node policy before signing.

Show the complete funding arrangement.

The wallet distinguishes BTC outputs, the NFT recipient script, fee and change. Any sub-minimum remainder's disposition remains visible. Additional funding outputs remain subject to Bitcoin transaction rules.

Fee changes require an appropriately funded arrangement and matching signatures. Existing signed transactions may still be broadcast, so a replaced preview does not revoke prior authorization.

Prove the identity. Authorize the spend.

Message signatures and funding signatures have separate purposes.

Identity proof.

The reference integration uses BIP322 message signatures [4] with an explicit action, expiry and network context. The signing service verifies address control before authorizing a holder action. NFT ownership is recorded as a supported recipient script; user-facing addresses are its presentation form.

Funding proof.

The funding address may differ from the address that owns the NFT. A separate ownership proof binds the funding address to the same operation and holder before the service reserves its inputs. Publicly visible coins and keys alone do not prove control. Message proof does not spend BTC.

Transaction authorization.

The wallet reviews a PSBT, as defined by BIP174 [2], then signs its own funding inputs. The preview distinguishes the NFT recipient, payment outputs, miner fee and change. The issuer's Seal signature is a separate authorization for the asset event.

Safe coin selection.

The reference integration prefers asset-aware pure-BTC candidates, excluding outputs identified as carrying other protocol assets. Bitcoin Core rechecks their scripts, values, unspent status and confirmations; coinbase inputs must be mature. Inputs need at least one confirmation under the service profile.

If safe candidate data is unavailable or inconsistent, coin selection pauses to avoid spending an unverified output. The reference adapters support UniSat, Xverse and OKX Wallet signing interfaces. Wallet signing support is distinct from native display or interpretation of this custom NFT registry.

Open issuance. Preserved continuity.

Supply control and service availability have different effects.

Supply closure.

A valid `CLOSE_MINT` event permanently sets `issuanceClosed`. Later `MINT` events are rejected so the collection cannot extend issuance after closure. Existing active assets can still `MOVE` or `RETIRE` under their predecessor rules.

State continuity.

`UPGRADE` supports an existing authority-anchored registry. It commits the prior checkpoint and accepted rules, proves the original issuer signature, and consumes that collection's controlling output as input 0. Exactly one authority-bearing collection is consumed, preventing unrelated registries from being combined.

Matching treasury outputs return at least 546 sats of authority value. The operation preserves collection identity, supply, content commitments, owner scripts, asset status and latest event references. It initializes per-asset versions to one and removes the global authority reference so later actions follow per-asset state.

Application pause.

A pause stops new service authorization and automatic processing. A draft without an issued Seal permit can release reservations. Once a valid permit is exported, its reservations remain tracked because the holder may submit the transaction outside the service; an expiry or pause cannot revoke the signature.

Before resuming, the operator reconciles pending transactions, issued permits, asset reservations and the effective chain. Availability controls therefore manage the signing service, while `CLOSE_MINT` permanently changes collection issuance state.

Explicit assumptions. Inspectable records.

Verification is meaningful when its trust boundary is clear.

Property	Dependency and limit
BTC spending authorization	Funding signatures and Bitcoin consensus protect the actual BTC inputs.
NFT event authorization	Issuer key security and valid Seals control registry transitions.
Holder consent	The signing service verifies holder intent; the event indexer does not independently require a holder proof.
Unique serial history	Accepted rules, operation uniqueness and exact predecessor/version checks prevent invalid state updates.
Content integrity	Hashes detect changed files when the catalog and content are available.
Registry consistency	Effective-chain replay and matching rules produce comparable checkpoints.

An issuer can authorize an asset transition with its own valid Seal. Independent verification authenticates that authorization, but does not establish that the service correctly checked holder intent. Integrators assess and secure that issuer boundary explicitly.

The protocol's NFT rules execute in application verifiers. Bitcoin consensus does not enforce this registry as a covenant. Transaction and identity data remain public Bitcoin information; protocol authorization does not add shielded-address privacy.

The rules hash pins exact file and dependency bytes. A different encoding, domain string or evaluator can produce a different registry. Interoperability requires identical interpretation and signature context, plus access to the relevant event and content data.

APPENDIX A / REFERENCE PARAMETERS

Reference parameters.

The following limits describe the selected reference rules.

Parameter	Reference value	Purpose
Supply bound	1 to 4,000	Caps serials accepted by this codec.
Batch entries	1 to 100	Bounds asset-event payloads.
Funding inputs	1 to 200, unique	Rejects duplicate outpoints and excessive transaction shapes.
Outputs	1 to 10	Bounds the committed transaction view.
Command output	One, zero-valued, index 0	Fixes unambiguous event location.
Recipient scripts	P2TR: 34 bytes; P2WPKH: 22	Defines supported ownership encodings.
Treasury script	P2TR: 34 bytes	Defines the deployment receiving script.

Event	Body after the common header
DEPLOY	u16 supply; catalog, profile, rules and issuer fields (32 bytes each); treasury script (34 bytes).
MINT	u8 recipient length; recipient; u8 count; serial:u16 for each entry.
MOVE / RETIRE	Recipient length and script; count; serial:u16, previousTxid:32, version:u32 for each entry. RETIRE has no recipient bytes.
CLOSE_MINT	Empty body.
UPGRADE	Previous state hash (32 bytes) and accepted rules hash (32 bytes).

All numeric fields are unsigned big-endian. Every payload ends with the 64-byte issuer signature. In a MOVE or RETIRE, a single entry occupies 38 bytes, supporting precise size estimates before signing.

APPENDIX B / TERMINOLOGY

Terms with one precise meaning.

A shared vocabulary for implementers and verifiers.

Term	Meaning
Collection ID	Network genesis hash and the original deployment transaction ID.
Serial	A fixed asset identifier within its collection.
Owner script	The supported receiving script recorded as NFT ownership.
Seal	Issuer authorization binding an event and the selected transaction view.
Operation ID	A 16-byte identifier accepted once within its collection.
Asset version	A counter advanced by an accepted MOVE or RETIRE.
Rules hash	A content hash that identifies exact reference rules and dependencies.
State hash	A deterministic checkpoint of one reconstructed collection.
UTXO	A Bitcoin output that has not been spent and can fund a transaction.
PSBT	An exchange format for transaction and input-signing data.
Retirement	Permanent circulation termination with the serial and history retained.

For payload sizing, N is the entry count and L is the recipient script length. Including the issuer signature and excluding OP_RETURN script framing: MINT uses $120 + L + 2N$ bytes; MOVE uses $120 + L + 38N$; RETIRE uses $120 + 38N$. DEPLOY uses 282, CLOSE_MINT 118 and UPGRADE 182 bytes.

A payload that fits the protocol bounds must still satisfy the broadcasting node's relay policy. The batch ceiling is not a guarantee that every node will relay the maximum-size transaction.

APPENDIX C / REFERENCES

Established methods. Defined NFT semantics.

The underlying Bitcoin methods retain their own specifications.

01 BIP340

Schnorr Signatures for secp256k1. The issuer Seal's signature foundation.

bips.dev/340/

02 BIP174

Partially Signed Bitcoin Transaction Format. The wallet-signing exchange format.

bips.dev/174/

03 Bitcoin Core RPC

estimatesmartfee. Dynamic rate estimation for the reference settlement service.

bitcoincore.org/en/doc/30.0.0/rpc/util/estimatesmartfee/

04 BIP322

Generic Signed Message Format. Address-control proofs in the integration profile.

bips.dev/322/

Normative identity and compatibility.

The selected content-addressed reference rules define exact framing, domain bytes, canonical serialization and executable transitions. Verifiers reproduce the rules hash from the ordered file digests and dependency manifest, then evaluate events under that identity. A display title is not a replacement for those bytes.

Document scope.

Version 1.0 is the public edition of this CRC-based NFT protocol whitepaper. It describes the protocol and reference integration, without prescribing collection content or commercial prices. Publication does not prove a particular deployment, asset issuance or live-service activation.

Protocol event validity, Bitcoin transaction validity and service completion are evaluated separately. This separation gives an implementer a clear basis for reconstructing the asset registry and reviewing the funds used to settle each operation.