Organic Bitcoin (OBTC) Whitepaper V1.1

Edition: Implementation-Aligned, Extended Readable Edition
Version: 1.1
Status: Request for Comments (RFC)
Audience: protocol engineers, wallet developers, mining pools, exchanges, researchers, operations teams, and community builders
Mainnet parameter notice
The current candidate mainnet fork height is `1000000`. This value is provisional and may change before final activation or before final mainnet-candidate release artifacts are published.
Under the current staged activation model, mainnet replay protection activates at `1000001`, calculated as `fork height + 1`. Expiry enforcement, canonical REAP hardening, and the expiry-state commitment activate at `1002016`, calculated as `fork height + 2016`.
Implementation source of truth: `obtcd/chaincfg/params_obtc.go`.
The implementation values in this edition were checked against `obtcd` commit:
`cd9ae639500bbffd82a8b42b1c6ca1c0152c629d`

---

0. Reading guide

The goal of this whitepaper is to **make the system clear**, not to bury the reader in jargon or present a speculative manifesto detached from implementation.
To lower the reading barrier, the suggested order is:

- **3-minute version**: Section 1, Section 2, Section 5.1
- **Product and operations version**: Section 6 (lifecycle), Section 11 (wallet), Section 14 (operations)
- **Engineering version**: Section 7 (index), Section 8 (REAP), Section 9 (replay protection), Section 13 (security)
- **Economics and governance version**: Section 12 (economics), Section 15 (governance)

This document uses English technical terms directly, such as UTXO, Replay Protection, Mempool, and Coinbase.
Each of them is also explained in plain language in **Section 19 (Glossary)**.

---

1. OBTC in one sentence

OBTC is a Bitcoin-derived monetary system that gives every UTXO a lifecycle:

- holders can spend or renew before expiry;
- after expiry, the ordinary owner-signature path closes;
- miners may process the expired output through **REAP (Reclaim Expired Assets Protocol)**;
- for a normal expired output, **70%** is recreated under the original locking script as a refund and **30%** is credited to the PoW miner reward as protocol tax;
- very small expired outputs below the fixed dust threshold are fully folded into tax instead of creating an uneconomic refund output;
- deterministic ordering, bounded input counts, replay protection, expiry indexing, and wallet-side renewal tooling make the lifecycle executable rather than merely conceptual.

From a user point of view, the rule is:

> Your asset remains under your control while it is actively maintained. If it remains dormant beyond the lifecycle boundary, the protocol moves it into a deterministic reclamation path.

1.1 Why it is called Organic

"Organic Bitcoin" does not mean "organic food Bitcoin."
It means **a Bitcoin-derived system with metabolism**.
A conventional UTXO can remain unchanged forever. OBTC instead models a lifecycle:
`creation -> use -> dormancy -> expiry -> reclaim -> refund regeneration or dust extinction`
The word "organic" points to three properties:

1. **lifecycle**: UTXOs have a defined age boundary;
2. **recycling**: dormant value can contribute to continuing network security;
3. **regeneration**: a non-dust refund becomes a new UTXO with a new lifecycle.

The distinction should not be overstated. Bitcoin's indefinite ownership model is a deliberate and powerful social contract. OBTC explores a different contract: long-term ownership remains available, but it requires occasional cryptographic activity.
REAP also needs to be described precisely. A normal REAP event does not automatically reduce the number of UTXOs: it consumes one old UTXO and usually creates one refund UTXO. UTXO-count reduction happens when:

- an expired input is below the dust threshold and creates no refund;
- a holder renews by consolidating several outputs into fewer outputs;
- normal wallet activity reorganizes fragmented state.

Therefore, OBTC's metabolism concerns both **value lifecycle** and **state hygiene**, but the two are not identical.

1.2 Where the innovation sits

Many underlying primitives are known:

- height-based age conditions,
- deterministic transaction construction,
- UTXO indexes,
- OP_RETURN markers and state commitments,
- block-template resource limits,
- wallet scheduling and renewal automation.

The innovation is primarily the way these pieces are combined into one closed monetary mechanism:

> If long-lived on-chain state is not free, and some claims will become permanently inaccessible, can a deterministic lifecycle preserve active ownership while gradually converting abandoned value into network security?

OBTC answers that question with an explicit protocol rather than an informal future policy.

---

2. Why build this at all

OBTC does not treat inactivity as morally wrong, and it does not claim that every old UTXO is lost. A long-unmoved output may represent deliberate cold storage, an inaccessible key, an estate, an abandoned wallet, or simply a holder who prefers not to transact.
The protocol cannot observe those private facts. It observes only **cryptographic activity**.
That distinction is fundamental:

> Dormancy is observable; permanent loss is not.

OBTC therefore does not attempt to classify individual owners or infer intent. It defines a public lifecycle rule that applies equally to all UTXOs and gives every active holder a long renewal path before protocol consequences apply.

2.1 The long-run structural problem

A Bitcoin-like UTXO system faces several long-horizon pressures:

- some supply becomes practically inaccessible through lost keys, broken backups, inaccessible estates, or operational failure;
- dormant outputs remain part of the live state even when their economic owners can no longer act;
- fragmented and tiny outputs impose indexing, scanning, and operational costs;
- subsidy declines over time, increasing dependence on fees and other miner-revenue sources;

- users and custodians have no protocol-level requirement to demonstrate that long-term claims remain operationally controlled.

These pressures are related but should not be confused. Historical block storage, current UTXO state, miner security budgets, and user key management are separate problems. OBTC links them through one lifecycle architecture without claiming that one mechanism eliminates every cost.

2.2 OBTC's answer

OBTC closes the loop through four components:

1. **Expiry** gives each UTXO a deterministic lifecycle boundary.
2. **Renewal** lets an active holder reset that boundary through an ordinary signed transaction.
3. **REAP** gives miners a tightly constrained system path for processing expired outputs.
4. **Refund plus tax** preserves most value under the original script while directing a fixed share into the PoW security budget.

The components justify one another:

- expiry without renewal would be punitive;
- renewal without protocol consequences would not address permanent dormancy;
- reclaim without refund would be equivalent to total seizure;
- refund without dust handling would regenerate uneconomic state indefinitely;
- miner tax without deterministic ordering would create excessive discretion;
- lifecycle rules without wallet tooling would impose unreasonable operational risk on users.

2.3 Design targets

- a clear and deterministic UTXO lifecycle;
- a long renewal horizon for active holders;
- gradual, finite decay of permanently dormant value;
- an endogenous PoW security-budget contribution;
- canonical and independently verifiable REAP construction;
- bounded block-template resource use;
- explicit distinction between consensus rules and wallet/mining policy;
- practical observability for users, miners, and node operators.

2.4 Non-goals

- proving that a specific dormant UTXO is permanently lost;
- preserving a model in which holders can ignore maintenance forever;

- guaranteeing a fixed amount of miner income in any block or year;
- requiring every node to store every historical block body forever;
- using discretionary committees to decide which outputs are reclaimed;
- not making any price, liquidity, exchange-listing, or adoption claims.

---

3. System overview: a three-layer architecture

OBTC can be understood as three layers with different responsibilities.
This separation is not just an explanatory convenience. It is how the system avoids conceptual confusion. One of the easiest ways to misunderstand OBTC is to mix up three different questions:

1. what the chain must accept or reject,
2. how miners and nodes should construct or prioritize valid behavior,
3. how users and automation systems should observe risk and act before protocol penalties matter.

OBTC draws those boundaries explicitly so that consensus remains strict, mining policy remains practical, and wallet/product logic remains flexible.

3.1 Consensus Layer

This layer answers: **what is a valid transaction and a valid block**.
Core responsibilities:

- expiry determination,
- REAP validity rules,
- replay-protection enforcement after activation.

Consensus is where ambiguity has to die. Different node implementations must converge here even when operators disagree, hardware differs, or blocks arrive under adversarial conditions. That is why expiry is expressed as a height-based rule rather than as a vague "long enough" concept, and why REAP validity is tied to marker checks, ordering constraints, and accounting invariants.
Consensus is also where the system draws hard boundaries around what users may no longer do. An expired UTXO is not merely "risky." After the relevant activation, it is no longer spendable through the ordinary path. Likewise, replay protection is not merely an ergonomic recommendation after activation; it becomes a direct validity requirement.

3.2 Mining and Policy Layer

This layer answers: **within the valid set, how should blocks be assembled safely**.
Core responsibilities:

- REAP candidate selection and construction,

- reserving weight budget for REAP in block templates,
- separating system transactions from ordinary mempool policy.

This layer matters because valid behavior is still not the same thing as safe or stable behavior. A miner needs a deterministic way to discover expired candidates, construct REAP transactions, respect block-size constraints, and keep system transactions separate from ordinary user transaction flow.
Policy is also where anti-MEV and anti-abuse concerns become practical. Canonical ordering, input ceilings, reserved block weight, and mempool exclusion all reduce the surface area for strategic manipulation without having to invent a broad new governance layer. In other words, the mining/policy layer exists to make the reclaim path executable under real block-building constraints.

3.3 Wallet Execution Layer

This layer answers: **how users discover risk and take action**.
Core capabilities:

- `obtc.getexpiry` : inspect expiry risk,
- `obtc.renew` : renew manually,
- `renewall` : renew in batches,
- Auto-Renew: automated renewal with failure backoff and budget ceilings.

This layer is where OBTC becomes usable rather than merely enforceable. A lifecycle rule without lifecycle tooling would shift too much burden onto memory, manual calendars, or after-the-fact crisis response. The wallet layer turns the protocol rule into actionable information: what is safe, what is expiring, what is already expired, and what the next low-risk action should be.
The wallet layer is also where controlled automation becomes realistic. Capability-based authorization, staged execution, scheduler limits, and audit trails do not belong in consensus. They belong in the execution environment that helps active users, operators, and wallet software stay ahead of expiry.

---

4. The core mechanism in plain language

Assume you hold a `1 OBTC` UTXO.

1. It is created at height `h_create` .
2. It becomes expired at `h_create + W` , where `W` is the network expiry window.
3. Before expiry, you can spend it normally or renew it to a new output.
4. After expiry, the ordinary spending path is no longer valid.
5. A miner may include that expired output in a valid REAP system transaction.
6. For a normal input: - `0.3 OBTC` becomes protocol tax credited through coinbase accounting; - `0.7 OBTC` is recreated under the original locking script; - the refund is a new UTXO with a new expiry height.

Three timing distinctions matter.

Expiry is not the same as immediate processing

At the expiry height, the UTXO becomes eligible for the system path. The refund does not exist until a miner actually includes the corresponding canonical REAP prefix in a block.

REAP inclusion is optional at consensus

A block without a REAP transaction can still be valid. This protects chain liveness when a miner has not integrated REAP construction or when a local template-building error occurs. The consequence is that the protocol does not guarantee an exact block in which a particular expired UTXO will be processed.

REAP contents are strict when present

If a miner includes REAP, the transaction is not discretionary. Its inputs must match the canonical global prefix, obey input caps, satisfy marker checks, preserve value accounting, and remain within normal block validity limits.
The user-facing interpretation is therefore:

> Renew before expiry to keep immediate control. After expiry, the UTXO enters a deterministic but capacity-limited reclamation queue.

---

5. Parameters and activation schedule (current implementation values)

OBTC separates permanent lifecycle parameters from launch and rollout parameters.

- **Lifecycle parameters** define UTXO age, tax, refund, dust behavior, ordering, and input ceilings.
- **Activation heights** define when already-implemented rules become mandatory on a particular network.
- **Mining and wallet defaults** guide software behavior but are not necessarily consensus rules.

This separation keeps operational rollout choices from being mistaken for the economic model itself.

5.1 Activation matrix across the three networks

Network	Fork Height	Expiry Index Start	Expiry Enable	Expiry Commitment	REAP Hardening	Replay Protection
Mainnet	1,000,000	0	1,002,016	1,002,016	1,002,016	1,000,001
Testnet	0	0	100	100	120	130
Regtest	100	0	110	110	112	114

Definitions:

- **Fork Height**: the height where OBTC enters its own rule domain.
- **Expiry Index Start**: the height from which expiry-related state is indexed.
- **Expiry Enable**: the point where ordinary spends of expired outputs become invalid.
- **Expiry Commitment**: the point where the coinbase expiry-state commitment becomes mandatory.
- **REAP Hardening**: the point where canonical REAP ordering and input caps become consensus-enforced.
- **Replay Protection**: the point where OBTC-specific signature-domain semantics become mandatory.

The mainnet fork height remains provisional.
Replay protection activates at `fork + 1` so the independent chain does not expose a post-fork transaction-replay window. Expiry and REAP hardening activate at `fork + 2016`, one standard 2,016-block interval later.
The `2016`-block delay is an operational launch boundary, not part of the long-term asset lifecycle. Public communication, wallet preparation, testnet rehearsal, and miner integration are expected to occur before the fork. The post-fork interval provides a fixed, auditable transition before expiry enforcement becomes mandatory.

5.2 Expiry horizon

Network	WindowBlocks	Approximate time at 10-minute blocks	ListBatchLimit
Mainnet	362,880	6.904 years	10,000
Testnet	144	1 day	5,000
Regtest	144	1 day	1,000

The mainnet value has three useful identities:

$$362{,}880 = 9! = 180 \times 2{,}016$$

At the nominal rate of `52,560` blocks per year:

$$T = \frac{362{,}880}{52{,}560} \approx 6.9041 \text{ years}$$

An exact nominal seven-year window would be `367,920` blocks. The difference is `5,040` blocks, approximately `35` days.
The factorial identity is not a proof of economic optimality. It explains the parameter's provenance and makes it easy to recognize. Its operational properties are more important:

- it is close to seven years;
- it is exactly `180` standard 2,016-block intervals;
- it gives holders a long maintenance horizon;
- it allows test networks to use much shorter windows without changing lifecycle semantics.

Expiry is calculated only from block height:

$$h_{expiry} = h_{create} + 362{,}880$$

No wall-clock oracle, price input, fee oracle, or governance vote is required.

5.3 Namespace Isolation parameters

- Network magic:
- Main: `0x4F425443`
- Test: `0x4F544553`
- Reg: `0x4F524547`
- Default P2P port: `9527 / 19527 / 29527`
- Default node RPC port: `9528 / 19528 / 29528`
- Default wallet legacy RPC port: `9554 / 19554 / 29554`
- Bech32 HRP: `obtc / obtct / obtcrt`
- isolated P2PKH, P2SH, witness, and private-key prefixes
- isolated HD private/public versions
- BIP44 coin type: `20260 / 20261 / 20262`

Implementation requirement: namespace collisions are checked at startup, and a node must refuse to start if a conflict is detected.
Namespace isolation and replay protection solve different problems:

- namespace isolation reduces accidental chain confusion;
- replay-protected signature domains prevent a valid signature on one chain from being valid on the other after activation.

Both are required.

5.4 REAP parameters and why they are separated by layer

Parameter	Mainnet value	Layer
Canonical sort mode	`Strict`	consensus after hardening
Normal input cap	`256`	consensus
Refundless dust input cap	`1,024`	consensus
Tax ratio	`30 / 100`	consensus
Refund ratio	`70 / 100`	consensus
Dust threshold	`720 sat`	consensus
REAP max weight (`ReapMaxWeight`)	`400,000 WU`	consensus
Maximum block weight	`4,000,000 WU`	consensus

Why 30% tax and 70% refund

For a normal expired input of value `v` :

$$Tax(v) = \left\lfloor \frac{30v}{100} \right\rfloor$$

$$Refund(v) = v - Tax(v)$$

If the holder never renews the refund, the value follows an approximately geometric path with retention ratio `0.7` per lifecycle.
The value half-life is:

$$n_{1/2} = \frac{\ln(0.5)}{\ln(0.7)} \approx 1.943 \text{ lifecycles}$$

At `6.9041` years per lifecycle, dormant value halves approximately every `13.4` years.
The 30% rate is intended as a middle path:

- lower rates preserve more value at the first event but allow inaccessible lineages to survive for much longer;
- higher rates clear dormant value faster but impose a sharper penalty on users who fail to renew;
- 30% keeps a substantial 70% refund while still producing finite dust extinction on a multi-century, rather than effectively permanent, horizon.

For an isolated `1 OBTC = 100,000,000 sat` lineage and a `720 sat` dust threshold, assuming prompt processing at every expiry and no renewal or merging:

Expiry tax	Refund ratio	Lifecycle of first refund below 720 sat	Final extinction lifecycle	Approximate years to extinction
20%	80%	54	55	379.7
25%	75%	42	43	296.9
30%	**70%**	**34**	**35**	**241.6**
40%	60%	24	25	172.6
50%	50%	18	19	131.2

The table does not claim that every expired output is processed immediately. Queue delay and optional REAP inclusion can only extend calendar time. Its purpose is to show the intrinsic decay of one permanently dormant lineage.

Why 720 sat

The threshold is fixed at:

$$720 = 6!$$

The factorial identity makes the constant easy to recognize, while the protocol purpose is practical: below this input value, creating another refund output would preserve extremely small state for another full lifecycle. The fixed threshold avoids fee-oracle and governance complexity.
The dust rule checks the **expired input value**, not the calculated refund value:

- if `0 < v < 720`, the full input becomes tax and no refund is created;
- if `v >= 720`, the normal 30/70 calculation applies, even if the resulting refund falls below `720`;
- that sub-threshold refund is removed at its next expiry event.

Why 256 normal inputs and 1,024 dust inputs

`256` is a compact power-of-two ceiling for normal inputs, each of which may require a refund output and therefore adds both input and output weight.
`1,024` is four times larger because dust-tier inputs create no refund outputs. A higher dust ceiling lets the system remove more small state without granting unlimited block-template growth.
The two limits are independent tiers, but selection remains one canonical prefix. If either applicable tier ceiling is reached, the transaction stops; miners cannot skip forward to improve the mix.

Why a 400,000-WU mainnet REAP max weight

The current mainnet consensus limit is `ReapMaxWeight = 400,000 WU`, equal to 10% of the `4,000,000-WU` block maximum. A block is invalid if a single REAP transaction exceeds this limit. Mining template construction uses this value as the default REAP weight budget and trims trailing canonical inputs until the constructed REAP transaction fits, while preserving prefix order. Changing the mainnet max weight is consensus-sensitive because validators enforce it directly.

5.5 Default Auto-Renew parameters

- Enabled: `false`
- Interval: `30m`
- Failure backoff: `15m` (may be `0`)
- Window: `window_end <= blocks_to_expiry <= window_start`
- `window_start = 52,560`
- `window_end = 25,920`
- MaxUtxosPerRun: `100`
- MaxFeeRate: `5,000 sat/KB`
- MaxRenewAmountPerRun: `0` (unlimited)
- `autorenewamount`: when enabled, it must be greater than `0`

These are wallet execution defaults, not consensus rules.
Auto-Renew is off by default. The scheduler operates within a bounded pre-expiry window, applies fee and candidate ceilings, and backs off after failure. The aim is to reduce user inaction without turning automation into uncontrolled spending.

---

6. UTXO lifecycle

The UTXO lifecycle is the conceptual center of OBTC.

6.1 The key formula

For any UTXO created at height `h_create`:

$$h_{expiry} = h_{create} + W$$

On mainnet:

$$W = 362{,}880$$

The UTXO becomes expired when:

$$h_{tip} \geq h_{expiry}$$

The rule is tied to the individual output, not to an address, wallet, or account. One wallet may therefore hold outputs with many different expiry heights.

6.2 Wallet-side status layers

Wallets may expose three user-facing states:

- `ok`: comfortably before expiry;
- `expiring`: within the configured warning or renewal window;
- `expired`: ordinary spending is no longer valid.

Only the final boundary is consensus. `ok` and `expiring` are wallet presentation and automation layers.

6.3 The two lifecycle paths

Active-holder path

Before expiry, the holder spends or renews the UTXO with a valid signature. The original output is consumed and one or more new outputs receive new creation heights.

System reclamation path

After expiry, miners may process the output through REAP. Ordinary spending is rejected. A normal input produces a tax plus a refund; a dust-tier input produces tax only.
This asymmetry is intentional. Renewal is the expected path for controlled assets. Reclamation is the backstop for long-term dormancy.

6.4 A full example

Input value:
`10,000,000 sat` (`0.1 OBTC`)
At REAP:

$$Tax = \lfloor 10{,}000{,}000 \times 0.30 \rfloor = 3{,}000{,}000 \text{ sat}$$

$$Refund = 10{,}000{,}000 - 3{,}000{,}000 = 7{,}000{,}000 \text{ sat}$$

The `7,000,000 sat` refund is created under the exact original `scriptPubKey` and receives a new creation height.
If that refund is also never renewed, the same rule applies after another `362,880` blocks. The sequence begins:
`10,000,000 -> 7,000,000 -> 4,900,000 -> 3,430,000 -> ...`
Integer rounding always preserves value because the tax is floored and the refund receives the remainder.

6.5 What happens to UTXO state

A lifecycle event can affect UTXO count in different ways:

Event	Inputs consumed	Outputs created	Typical UTXO-count effect
Normal spend	one or more	one or more	depends on transaction
Renewal without consolidation	1	1	unchanged
Renewal with consolidation	many	fewer	decreases
Normal REAP	1	1 refund	unchanged
Dust-fold REAP	1	0 refund	decreases by 1

Therefore, REAP gradually removes inaccessible **value**, but only dust folding and consolidation directly reduce UTXO count. Historical block pruning is a separate storage concern.

---

7. Expiry Index: how to track expiring assets efficiently

The lifecycle rule is easy to state and expensive to enforce naively. The chain cannot afford to rediscover expiry from scratch by repeatedly scanning the full live set. That is why expiry indexing is not an optional optimization. It is a structural requirement for making reclaim operationally feasible.

7.1 Why an index is mandatory

It is not acceptable to scan the entire historical set on every new block just to ask "what expired now?" That is why a dedicated expiry index is required.

The reason is both obvious and fundamental. The UTXO set can be large, while the number of outputs that become relevant at a given height may be comparatively small. A protocol that required full-state scanning at every block just to identify candidates would turn a logically simple rule into an operational burden on every node and miner.

So OBTC shifts work from repeated global scanning into incremental maintenance. Expiry is computed once when outputs are created, persisted in an index, updated when outputs are spent, and scanned in ordered form only when needed. That is the difference between a lifecycle model that sounds elegant in prose and one that actually fits block-by-block operation.

7.2 Index structures and canonical scan keys

The current implementation maintains several related mappings because lifecycle queries and consensus-prefix validation need different orderings:

1. `OutPoint -> ExpiryKey` - supports fast lookup and deletion when an output is spent or renewed;
2. `(ExpiryKey || OrderedOutPoint) -> empty` - supports general expiry-range scans and resumable wallet/operator queries;
3. `(ExpiryKey || Amount || ReapOrderedOutPoint) -> empty` - supports the strict canonical REAP prefix;
4. `OutPoint -> ReapStrictCompositeKey` - supports fast deletion from the strict candidate index.

The general range index and strict REAP index must not be confused.

- general expiry scans are ordered by expiry and an outpoint encoding suitable for paged queries;
- consensus REAP scans are ordered by expiry, amount, raw transaction-hash bytes, and numeric output index.

Integer fields are encoded big-endian where lexicographic database order must match numeric order.
The strict outpoint encoding uses raw internal hash bytes followed by big-endian `vout`, matching the consensus comparator.
The index stores one composite-key entry per live UTXO rather than one large list value per expiry height. This keeps updates local, avoids rewriting large bucket values, and allows bounded prefix scans directly from database order.

7.3 Block connection and rollback

- `ConnectBlock` : add new outputs into the index, remove spent inputs from the index;
- `DisconnectBlock` : restore the reverse state.

That keeps the index aligned with the active chain even under reorg.
The core discipline here is symmetry. Every live update path must have a rollback path that reconstructs the same indexed state the node would have had if the disconnected block had never been accepted. Without that property, expiry indexing would drift under reorganizations and eventually stop being a reliable source of reclaim candidates.
This is also why the index tracks its own notion of indexed tip height. Nodes need to know not only that buckets exist, but also what chain state those buckets correspond to. A correct expiry index is not merely a set of key-value pairs; it is a state summary attached to a particular active-chain tip.
Rebuild exists because live maintenance is not the only correctness path. If the index is missing, version-mismatched, corrupted, or explicitly reindexed, the node must be able to reconstruct expiry state from the underlying UTXO set. The correctness target is that live maintenance and rebuild arrive at the same effective indexed state for the same active chain.

7.4 Paged scanning and resumable queries

Suggested interface:
`(fromKey, toKey, maxResults, startAfter)`

- `maxResults` controls page size,
- `startAfter` supports resumable scans,
- suitable for large queries without returning excessive result sets in one response.

This is not only an RPC ergonomics choice. It is also an operational safety choice. Large reclaim-related scans can be expensive or long-running if the expired inventory is large. Pagination prevents the observation and planning layer from assuming that the entire relevant set must be loaded or returned in one step.
Because the scan index is ordered, cursor-based continuation is stable enough to support both operators and automated systems. A caller can resume from a known `(height, outpoint)` boundary without re-scanning from genesis or relying on unstable list offsets.
In practice, that means the expiry index supports more than reclaim construction. It also supports observability, planning, dry-run analysis, and operational dashboards that need to reason about the expiring set in manageable chunks.

7.5 Expiry commitments

The implementation maintains an incremental expiry-state accumulator and commits its root through an OP_RETURN output in coinbase after activation.
Key properties:

- the coinbase at height `h` commits the indexed expiry pre-state corresponding to the parent tip at `h-1`;
- each post-activation block must contain exactly one correctly formatted commitment;
- the committed root must match the validator's locally calculated state;
- block connection, disconnection, rebuild, and snapshot-import paths must converge on the same state.

The commitment and the canonical REAP order serve different roles:

- the commitment anchors the **set of indexed expiry state**;
- explicit sorting and canonical-prefix validation enforce the **order of selected REAP inputs**.

A set commitment is not a substitute for validating transaction order. Full validators independently derive the expected prefix from their local expiry index and compare it input by input with the block's REAP transaction.
The `getexpirycommitment` RPC exposes the current commitment, indexed tip, activation height, and whether the next block must include a commitment.

8. REAP: the full rule set for system reclaim transactions

REAP is the mechanism that makes expiry economically and operationally consequential. Because it spends outputs without the ordinary owner-signature path, its validity surface is intentionally narrow.

8.1 What REAP is

REAP is a miner-constructed, block-internal system transaction that processes expired UTXOs.
It is not:

- an ordinary wallet transaction;
- a mempool auction;
- a discretionary transfer to arbitrary recipients;
- a right for miners to select whichever expired outputs they prefer.

A block may omit REAP and remain valid. When REAP is present, its structure, inputs, outputs, marker, accounting, and canonical prefix are consensus-validated.
The default block-template implementation attempts to build REAP when eligible candidates exist. If local construction fails, the miner can still produce an otherwise valid block. This design prioritizes chain liveness while leaving processing speed economically driven.

8.2 Minimum conditions for validity

A valid REAP transaction must satisfy at least the following:

- transaction version and marker identify the system transaction;
- every input references a live expired UTXO;
- an unconfirmed or non-expired input is invalid;
- no block contains more than one REAP transaction after hardening;
- inputs match the global canonical prefix;
- input counts respect normal and dust-tier ceilings;
- tax, refund, and dust calculations are exact;
- each non-dust refund returns to the original `scriptPubKey`;
- marker height, count, and digest match the transaction;
- coinbase does not claim more than the permitted subsidy, fees, and REAP tax.

8.3 Tax, refund, rounding, and dust

For an expired input of value `v >= 720`:

$$Tax(v) = \left\lfloor \frac{30v}{100} \right\rfloor$$

$$Refund(v) = v - Tax(v)$$

The refund output uses the exact original locking script.
For an expired input satisfying:

$$0 < v < 720$$

no refund output is created:

$$Tax(v) = v$$

$$Refund(v) = 0$$

The input-value test is deliberate. If an input is at least `720 sat`, the normal 30/70 split applies even when the resulting refund is below `720 sat`. That refund remains a UTXO for one more lifecycle and is then fully folded when it later expires below the threshold.

8.4 Marker binding

The marker format is:
`REAP:<height>:<count>:<digest>`
Consensus validates:

- `height` equals the containing block height;
- `count` equals the number of REAP inputs;
- `digest` equals the digest of the ordered input outpoints.

The marker makes the system transaction self-describing and binds its context to the block and input vector.

8.5 Canonical order and global-prefix enforcement

After hardening, REAP inputs follow this total order:

1. `expiry height ASC`
2. `amount ASC`
3. `outpoint ASC`

This must be interpreted correctly.
The amount sort is **cohort-local**. The entire historical backlog is not globally reordered by value. Earlier expiry heights always remain ahead of later expiry heights. Amount only orders UTXOs that share the same expiry height, which normally means outputs created at the same source height under a fixed window.
Within one equal-expiry cohort, the small-to-large amount order creates a deterministic reward path:

- lower-value state is processed first;
- larger-value outputs appear later in the same cohort;
- miners cannot jump directly to the largest outputs;
- all validating nodes can derive the same next prefix.

The final `outpoint` tie-break must have one exact encoding and byte order across implementations. Canonical order is consensus-critical. If a miner includes `k` REAP inputs, validators independently calculate the first `k` eligible candidates and require exact equality. Skipping, substituting, or reordering any prefix input makes the block invalid.

8.6 Two-tier input ceilings

Mainnet consensus limits are:

- normal inputs (`v >= 720`): at most `256` ;
- refundless dust inputs (`0 < v < 720`): at most `1,024` .

Selection still follows one global prefix. The tiers do not create two independent queues and do not permit cherry-picking.
If the next canonical candidate would exceed the applicable tier ceiling, the REAP transaction ends. A miner cannot skip that candidate and continue with a later one.

8.7 Weight budgeting

The current mainnet REAP max weight is `400,000 WU` . Validators reject any REAP transaction above this consensus limit.
The mining builder constructs the canonical plan and trims trailing inputs until the actual transaction fits the configured budget, while preserving prefix order. The default mainnet template budget follows

`ReapMaxWeight`, so template construction and consensus validation are aligned. The final block must still satisfy the normal maximum block weight and signature-operation rules.

8.8 Coinbase accounting

Let:

- `BaseSubsidy(h)` be the normal subsidy;
- `Fees(h)` be ordinary transaction fees;
- `ReapTax(h)` be the sum of tax from included REAP inputs.

Then:

$$CoinbaseValue(h) \leq BaseSubsidy(h) + Fees(h) + ReapTax(h)$$

REAP tax is therefore not a separate unbounded output. It is included in ordinary coinbase upper-bound validation and follows coinbase maturity.

8.9 Conservation invariant

For every REAP transaction:

$$\sum Inputs = \sum Refunds + \sum Tax$$

This invariant must hold exactly in satoshis.

8.10 Optional inclusion and processing liveness

REAP validity is strict, but inclusion is optional.
This design has clear benefits:

- a REAP planner failure does not halt block production;
- a miner can join the network before integrating the full template path;
- consensus does not require every implementation to use identical internal planning software;
- high tax value provides a direct economic reason to include valid REAP transactions.

It also has explicit costs:

- no consensus rule guarantees that every eligible output is processed within a fixed number of blocks;
- backlog-clearance estimates depend on miner inclusion behavior;
- an expired output remains pending and cannot be spent normally until its canonical prefix is processed;
- low-tax prefix regions may create weaker short-term incentives.

These are not hidden properties. Wallets and operators should expose pending-expired state, miners should monitor missed REAP opportunities, and public claims should distinguish **validity guarantees** from **processing-rate assumptions**.

---

9. Replay Protection: how cross-chain replay is prevented

Cross-chain safety is not just a signing detail. A forked system must defend against both accidental operator confusion and deliberate signature reuse. OBTC therefore treats replay protection as a layered system rather than as one isolated bit flag.

9.1 Two defensive layers

1. **Namespace Isolation** at the address, port, HD, and coin-type level.
2. **Replay-protected sighash domain** at the signature-message level.

These layers solve different problems. Namespace isolation makes it harder to connect to the wrong chain, derive addresses under the wrong assumptions, or send value to an address format that looks deceptively familiar. Replay-protected sighash semantics stop valid signatures on one chain from remaining valid on the other chain after activation.
If OBTC only used namespace isolation, cross-chain signature reuse would remain a risk. If it only used replay-protected signatures, users and tooling could still make preventable operational mistakes before signing. Using both closes both the ergonomic and cryptographic gaps.

9.2 Implementation details

- replay bit: `0x40`
- domain tags: separate tags for Legacy, Witness, and Taproot paths

The high-level point is that replay protection is not treated as a single monolithic script-path rule. Different signing modes need explicit domain separation so that replay isolation actually covers the transaction forms the system intends to support.
That is why the whitepaper describes replay protection as a signature-domain design rather than as a wallet warning. The system must ensure that the message being signed is chain-specific after activation, not merely that wallets try to be careful.

9.3 Activation gate

The configured activation height is network-specific. On mainnet, replay protection is enforced at `ObtcMainNetForkHeight + 1`, currently `1000001` with the provisional fork anchor at `1000000`. That is the first post-fork OBTC block, so replay-protected signatures are mandatory immediately in the independent OBTC chain.

- before the configured activation height: compatibility semantics may continue to work;
- from the configured activation height onward: any signature missing replay-protected semantics fails directly.

For mainnet, replay protection is therefore not a late rollout step after expiry or REAP. It is intentionally earlier than the `1002016` staged activation height for expiry enforcement, REAP hardening, and expiry commitment so that the fork does not create a BTC/OBTC replay window.
This also improves review clarity. Operators know when replay-protected semantics are merely recommended and when they are mandatory. That is better than a vague "replay protection is generally important" claim that leaves the actual enforcement moment ambiguous.

9.4 Path coverage

- Legacy
- SegWit v0
- Taproot, both key path and script path

Coverage matters because partial replay protection is not full replay protection. If some signature paths were isolated and others were not, the remaining gap would become the most interesting path for mistakes or abuse.
OBTC therefore states path coverage explicitly. The goal is not to say replay protection exists in spirit, but that the relevant signing families are brought under the same chain-specific isolation model.

10. Node operation model

The lifecycle model changes node responsibilities, but it does not rewrite history.

10.1 Pruned Full Node vs Archive Node

- **Pruned Full Node**: retains the live UTXO set, expiry-related state, all headers, and a rolling window of block bodies sufficient for configured operation.
- **Archive Node**: retains all historical blocks and transaction bodies.

A pruned full node remains a validating node. Archive nodes provide complete historical query and independent audit capacity.

10.2 Invariants

- accepted block history is never rewritten by pruning;
- pruning changes local retention, not consensus rules;
- the live UTXO set remains necessary;
- the expiry index must correspond to the active chain tip;
- `ConnectBlock` and `DisconnectBlock` must update and restore lifecycle state symmetrically;
- rebuild from the same active-chain state must converge on the same index and commitment.

10.3 What lifecycle processing does and does not remove

Three storage concepts must remain separate:

1. **historical block bodies**: may be pruned according to node policy;
2. **live UTXO entries**: remain until spent or removed through dust fold;
3. **expiry-index entries**: add operational metadata for lifecycle processing.

A normal REAP transaction usually replaces one UTXO with one refund UTXO. It does not immediately shrink the UTXO set.
State reduction comes primarily from:

- dust-fold REAP, which creates no refund;
- wallet renewal that consolidates many outputs;
- ordinary transaction activity that reduces fragmentation.

This distinction prevents storage claims from overstating what REAP alone accomplishes.

10.4 Snapshot synchronization and UTXO commitments

Snapshot-assisted sync can reduce time to participation, but it must not turn one publisher into a hidden trust authority.
A safe process should include:

- a snapshot tied to a specific height and block hash;
- integrity checks for the UTXO and expiry state;
- multiple independent publication sources;
- forward synchronization and validation from the snapshot point;
- eventual background or archival verification where operationally required.

Snapshots accelerate state acquisition. They do not change the validity rules for expiry, ordinary spending, REAP, or commitments.

---

11. Wallet capability closure

Protocol rules alone are not enough for a lifecycle system. Users need interfaces that turn expiry from an abstract property into an actionable operating picture. That is why the wallet layer in OBTC is not an optional convenience add-on; it is part of how the broader design remains usable.

11.1 `obtc.getexpiry`

Returns the key risk fields for expiry management:

- outpoint
- amount
- create height
- expiry height
- blocks_to_expiry / days_to_expiry
- status
- dust_risk

This RPC is the observation surface for lifecycle risk. It allows users, operators, and automation systems to ask not only "what do I own?" but "what is approaching a maintenance boundary?" That is a materially different question from ordinary balance reporting.
The inclusion of both height-based and user-facing fields matters. Height and blocks-to-expiry anchor the answer in protocol reality. Status and dust-risk style fields make the answer usable for wallets and dashboards. In effect, `obtc.getexpiry` is where the lifecycle model becomes inspectable rather than implicit.

11.2 `obtc.renew`

Supports explicit renewal by outpoint, with parameters including:

- amount
- target address (optional)
- max fee rate (optional)
- minconf (optional)

It returns a transaction summary such as txid, number of inputs and outputs, and fee rate.
The important design point is that renewal is an explicit first-class action, not an obscure side effect of some unrelated transaction path. A user or operator can deliberately choose which outputs to renew, under what fee assumptions, and toward which target address.

That directness matters for safety. If renewal is meant to be the rational active-user alternative to reclaim, then it has to be visible, auditable, and easy to reason about. `obtc.renew` is the narrow manual path that provides that clarity.

11.3 `renewall`

Supports:

- status or window-based filtering,
- dry-run mode,
- interval/runs scheduling.

`renewall` exists because lifecycle maintenance quickly becomes an operational problem rather than a one-output problem. Once users or operators manage many outputs, the relevant question is not "can I renew one UTXO?" but "can I review and manage renewal work as a bounded batch process?"
That is why filtering, dry-run behavior, and scheduling matter. The tool is meant to support repeatable maintenance with previewability, not just mass submission. In practice, it forms a bridge between one-off manual renewal and continuous automated policy execution.

11.4 Auto-Renew scheduler

Features:

- periodic execution,
- window-based filtering,
- candidate count limit,
- fee-rate ceiling,
- failure backoff,
- per-run budget control.

The goal is automated risk mitigation within explicit safety boundaries.
This is an important boundary in the design. Auto-Renew does not change consensus and does not hide lifecycle rules from the user. It sits above consensus as a wallet-side service that tries to keep attentive users from drifting into reclaimable state because of pure operational neglect.
The safety boundaries are part of the design, not a footnote. Candidate limits, fee ceilings, budgets, and backoff keep the scheduler from turning normal protective automation into runaway spending or repeated failure loops. Automation without these controls would weaken rather than strengthen the lifecycle model.

11.5 The wallet automation direction

OBTC has a natural extension path at the wallet layer: **make lifecycle asset management observable, previewable, and safe to automate**.
The reason is straightforward:

- expiry risk has to be monitored continuously, not remembered manually once every few years;
- renewal should be previewable before signing or broadcasting;
- batch scheduling, budgets, and failure backoff are operational controls, not consensus rules;
- audit trails are essential when renewal is delegated to software or repeated operator workflows.

That is why the interface and wallet layer should prioritize:

- capability-based authorization instead of all-or-nothing unlock semantics,
- a controlled pipeline of `preview -> approve -> sign -> publish`,
- separation between a watch-only planner and an isolated signer,
- event-driven alerts for expiry, failure, and policy conditions,
- default auditability through operation metadata and policy metadata.

The deeper point is that lifecycle asset management naturally produces tasks that software can monitor, schedule, preview, and record more reliably than humans can do purely from memory. That is an operations claim, not a consensus claim.
That is also why the wallet/product layer is the right place for this direction. The chain only needs deterministic validity rules. Planning, authorization, staged signing, and automation policies belong above the chain in systems that can evolve faster and preserve clearer operator control.
This is a **wallet and product-layer direction**, not a new consensus rule.

---

12. Economics and game theory

OBTC creates a different reward and maintenance structure from a system in which dormant outputs remain inert forever. The economics should be read as conditional mechanisms, not guaranteed forecasts.

12.1 Long-term security-budget transformation

Let:

- `T` be the lifecycle length in years;
- $\rho = 0.70$ be the retained refund fraction;
- $\tau = 0.30$ be the tax fraction;
- `L` be the fraction of supply that remains permanently dormant.

The equivalent continuous annual decay rate of that dormant subset is:

$$p = -\frac{\ln(\rho)}{T}$$

For:

$$T = \frac{362{,}880}{52{,}560} \approx 6.9041$$

and $\rho = 0.7$:

$$p \approx 5.166\% \text{ per year}$$

A rough annualized security-budget transformation is:

$$B \approx Lp$$

This expression is useful for comparing parameter combinations. It is not a promise of smooth yearly miner revenue because actual tax arrives in discrete blocks, depends on miner inclusion, and is shaped by the historical backlog.

12.2 Dormant-lineage decay and finite dust extinction

For a normal isolated lineage:

$$v_{n+1} = v_n - \lfloor 0.30 v_n \rfloor$$

which is equivalent to:

$$v_{n+1} = \lceil 0.70 v_n \rceil$$

until the input is below `720 sat`.
Ignoring integer rounding, the approximate number of lifecycle refunds needed to fall below dust is:

$$N_{dust} \approx \left\lceil \frac{\ln(D/V_0)}{\ln(\rho)} \right\rceil$$

where `D = 720` and `V_0` is the initial value.

Because the dust test applies to the next expired input, final extinction normally occurs one REAP event after the first refund falls below `720 sat`.
For `V_0 = 100,000,000 sat`:

- the 34th normal REAP refund is `543 sat`;
- the 35th REAP sees a dust-tier input and creates no refund;
- intrinsic extinction time is approximately `35 x 6.9041 = 241.6 years`.

This is intentionally gradual. A 30% tax does not erase large dormant value quickly; it places an upper structure on indefinite persistence while retaining 70% at every normal event.
Queue waiting and optional miner inclusion can only lengthen calendar extinction time.

12.3 Incentives for active holders

Let ϕ be renewal cost divided by UTXO value. A holder who controls the key has a direct reason to renew whenever:

$$\phi < \tau$$

For large UTXOs, transaction fees are normally much smaller than 30% of value, so renewal dominates passivity.
Small outputs face a different problem: transaction fees and operational overhead may approach the output value. This is why wallets need batching, consolidation, fee ceilings, and pre-expiry warnings rather than treating every output identically.

12.4 Miner incentives and cohort-local reward ordering

The canonical order combines two goals:

- oldest expiry first across the full queue;
- small-to-large value within one equal-expiry cohort.

The amount component does not let miners globally prioritize high-value UTXOs. Instead, it produces a monotonic reward path inside each historical cohort while requiring low-value prefix processing before later high-value items can be reached.
This structure may encourage continued mining through a cohort, but it does not guarantee that one miner who clears low-value prefix entries will later win the high-value blocks. Optional inclusion therefore leaves a possible public-good tension in very low-tax regions.
The protocol's response is not discretionary selection. It is transparent ordering, immediate tax reward for included inputs, and operational monitoring of omission behavior.

12.5 Immediate expired backlog from historical inheritance

OBTC inherits Bitcoin history rather than starting from an empty UTXO set.
Let:

- `F` be fork height;
- `A` be expiry activation height;
- `W` be the expiry window.

A historical UTXO is already expired at activation when:

$$h_{create} + W \leq A$$

For the current candidate values:

- `F = 1,000,000`;
- `A = 1,002,016`;
- `W = 362,880`;

therefore:

$$h_{create} \leq 639{,}136$$

All still-live outputs satisfying that condition become eligible when expiry enforcement activates. They do not fit into one block. They form a bounded, ordered backlog processed over many blocks under the input ceilings, template weight policy, and actual miner inclusion rate.
Renewal before activation can reduce this inherited backlog. Delaying activation also allows more recent historical outputs to age into the eligible range, so the relationship between delay and backlog size is not one-directional.

12.6 Offline historical preview

An offline preview was executed from a BTC historical shadow snapshot at height `953,697` with:

- `165,782,759` live spendable UTXOs;
- `30%` tax;
- `720 sat` dust threshold;
- `256` normal-input cap;
- `1,024` dust-input cap;
- a `400,000-WU` stress-test budget.

Under the extreme assumption of no proactive renewal or consolidation, the preview reported:

Metric	Result
Maximum pending backlog	33,095,769 inputs
First inherited backlog clear	approximately 3.31 years
Final preview REAP block	approximately 7.46 years after preview activation
Maximum estimated REAP weight	253,948 WU
Dust share by input count	44.40%
Dust share of total tax	approximately 0.0059%

Interpretation:

- the selector did not deadlock under the tested envelope;
- dust is primarily a state-processing load, not a major revenue source;
- historical processing is a multi-year mechanism, not an instant purge;
- reward distribution can be highly uneven across blocks;
- backlog time is conditional on continuous template inclusion.

The preview used a `400,000-WU` envelope, which now matches the current mainnet `ReapMaxWeight`. Exact live clearance projections still depend on the final snapshot, renewal behavior, consolidation, and actual miner inclusion rate, so the preview should be read as an implementation-aligned rehearsal rather than a production forecast.

12.7 What the economic model does not guarantee

The lifecycle mechanism does not guarantee:

- a particular market price;
- a fixed miner-revenue amount;
- immediate REAP processing;
- a fixed backlog-clearance date;
- that every dormant UTXO is truly abandoned;
- that network adoption follows automatically from additional miner reward.

It guarantees only the protocol accounting and validity rules. Economic outcomes depend on renewal behavior, miner participation, transaction demand, and ecosystem integration.

13. Security model and threat analysis

The security model spans consensus, mining policy, wallet behavior, synchronization, and operations.

13.1 Threats

- forged or malformed REAP markers;
- non-expired inputs in a REAP transaction;
- reordered, skipped, or substituted canonical-prefix inputs;
- multiple REAP transactions in one block after hardening;
- oversized input sets or block-resource exhaustion;
- coinbase over-claim of REAP tax;
- cross-chain transaction replay;
- fake system transactions injected into the mempool;
- expiry-index drift across connect, disconnect, rebuild, or snapshot paths;
- poisoned or inconsistent snapshots;
- reward spikes that increase short-range reorg incentives;
- prolonged REAP omission while expired outputs remain pending;
- wallet automation that retries, overspends, or acts outside user policy.

13.2 Defensive lines

- height-based expiry with no wall-clock oracle;
- exact marker height, count, and input-digest checks;
- script-neutral expired-input validation;
- canonical global-prefix enforcement;
- normal and dust-tier input caps;
- the mainnet REAP max weight and matching default template budget;
- exact value-conservation and coinbase upper-bound checks;
- coinbase maturity for REAP tax;
- mempool rejection of system transactions;
- namespace isolation plus signature-domain replay protection;
- symmetric index connect/disconnect behavior;
- expiry-state commitments;
- multi-source snapshot verification;
- default-off wallet automation with fee, count, budget, and retry controls;

- operational metrics for inclusion ratio, omission streaks, and pending backlog.

The governing principle is:

> A powerful system path must be narrow, deterministic, independently reproducible, and observable when it fails to make progress.

---

14. Operations and observability

Lifecycle consensus cannot be operated safely through balance and block-height metrics alone. Nodes, miners, and wallets must expose the state transitions that matter.

14.1 Suggested metrics

Consensus and index

- expired ordinary-spend rejection count;
- REAP non-expired-input rejection count;
- bad canonical-prefix rejection count;
- marker mismatch rejection count;
- replay-protection violation count;
- expiry commitment missing, duplicate, format, and mismatch counts;
- expiry-index tip versus active-chain tip;
- rebuild or rollback divergence count.

Mining and backlog

- eligible expired input count and value;
- oldest pending expiry height;
- template attempts with eligible candidates;
- planned REAP input count and tax;
- actual REAP input count and tax;
- REAP append success rate;
- blocks with candidates but no REAP;
- longest consecutive omission streak;
- normal and dust tier utilization;
- planned and actual REAP weight;
- REAP tax as a share of total coinbase value.

Wallet

- UTXOs by `ok`, `expiring`, and `expired` status;
- auto-renew candidates per run;
- manual and automated renewal success rate;
- fee-ceiling rejection count;
- per-run budget truncation count;
- failure-backoff activation count;
- outputs approaching dust risk;
- confirmation delay for renewal transactions.

14.2 Suggested alerts

- expiry index behind the active chain tip;
- commitment mismatch or missing commitment after activation;
- consecutive blocks with eligible candidates but no REAP above an operator-defined threshold;
- expired backlog growing faster than it is processed;
- oldest pending expiry age exceeding an operational threshold;
- sustained REAP planner failure;
- abnormal coinbase tax claim;
- repeated auto-renew failure or fee-ceiling rejection near expiry;
- wallet attempting ordinary spend of an expired output.

14.3 Validation and rehearsal

1. consensus tests for expiry, REAP, tax/refund arithmetic, dust fold, and replay protection;
2. canonical-prefix vectors shared across implementations;
3. connect/disconnect/rebuild tests for expiry state;
4. accelerated-expiry testnet rehearsals;
5. mining tests at zero, partial, and full tier utilization;
6. high-mempool-load tests with REAP weight reservation;
7. wallet tests for manual renewal, dry-run batching, scheduling, and backoff;
8. snapshot import and multi-source consistency tests;
9. long-running observation of backlog, omission streaks, and reward variance.

15. Governance and upgrades

OBTC minimizes governance by separating constants that define the monetary contract from defaults that guide software operation.

15.1 Consensus-sensitive parameters

Changes to the following alter chain validity or ownership expectations and therefore require explicit delayed activation and broad ecosystem review:

- expiry window;
- tax and refund ratio;
- dust threshold and dust-fold rule;
- canonical ordering and outpoint encoding;
- normal and dust input caps;
- expiry enforcement height;
- REAP hardening height;
- replay-protection rules;
- REAP max weight;
- expiry commitment rules;
- coinbase accounting.

15.2 Policy and wallet parameters

The following can generally evolve without changing the monetary contract, provided software remains within consensus limits:

- local REAP template reserve below the consensus max;
- scan batch sizes;
- warning windows;
- Auto-Renew schedule;
- fee ceilings;
- per-run renewal count and amount limits;
- monitoring and alert thresholds.

15.3 Upgrade principles

- changes must be mechanical and reproducible;
- activation heights must be explicit;

- users and operators need sufficient notice;
- parameter rationale and compatibility impact must be published;
- emergency discretion should be minimized;
- no committee receives authority to choose individual UTXOs or override canonical order.

---

16. Mathematical appendix

16.1 Lifecycle duration

With `Y = 52,560` nominal blocks per year:

$$T = \frac{W}{Y}$$

For `W = 362,880`:

$$T \approx 6.904109589 \text{ years}$$

16.2 Tax and refund recurrence

For normal input value `v_n >= D`:

$$t_n = \lfloor \tau v_n \rfloor$$

$$v_{n+1} = v_n - t_n = \lceil (1-\tau) v_n \rceil$$

For OBTC:

$$\tau = 0.30, \qquad \rho = 1 - \tau = 0.70, \qquad D = 720$$

For `0 < v_n < D`:

$$t_n = v_n, \qquad v_{n+1} = 0$$

16.3 Value half-life

$$n_{1/2} = \frac{\ln(0.5)}{\ln(\rho)}$$

For $\rho = 0.7$:

$$n_{1/2} \approx 1.943$$

16.4 Approximate dust-entry lifecycle

Ignoring integer rounding:

$$N_{dust} \approx \left\lceil \frac{\ln(D/V_0)}{\ln(\rho)} \right\rceil$$

Final extinction normally occurs at:

$$N_{extinct} = N_{dust} + 1$$

because the first sub-threshold refund remains live until its next expiry.

16.5 Annualized dormant-value decay

$$p = -\frac{\ln(\rho)}{T}$$

16.6 Approximate annual security-budget transformation

For permanently dormant supply share `L`:

$$B \approx Lp$$

This is an annualized comparison metric, not a block-by-block forecast.

16.7 Equal-decay parameter conversion

To preserve the same annualized decay `p` under a different lifecycle `T'`:

$$\rho' = e^{-pT'}$$

$$\tau' = 1 - \rho'$$

16.8 Queue-service lower bounds

Let:

- `N_n` be pending normal inputs;
- `N_d` be pending dust inputs;
- `C_n` and `C_d` be their per-transaction caps;
- `q` be the fraction of blocks that include a sufficiently full REAP transaction.

Ignoring weight interactions and new arrivals, a basic lower bound is:

$$Blocks \geq \frac{1}{q} \max\left(\left\lceil \frac{N_n}{C_n} \right\rceil, \left\lceil \frac{N_d}{C_d} \right\rceil\right)$$

Canonical-prefix composition, actual transaction weight, reward incentives, and new expiry arrivals can increase the realized time.

---

17. Key KPIs

Lifecycle and wallet

- supply expiring within 30, 90, and 365 days;
- renewal success rate;
- median and p95 renewal fee-to-value ratio;
- outputs renewed through consolidation;
- outputs entering expired state without prior warning delivery;
- Auto-Renew adoption and failure rate.

REAP and backlog

- eligible pending count and value;
- oldest pending expiry height;
- REAP inclusion ratio;
- longest omission streak;
- normal and dust inputs processed per block;
- backlog arrival rate versus service rate;
- first and final clearance estimates under observed inclusion;
- REAP weight utilization.

Economics and security

- REAP tax as a share of subsidy, fees, and total miner revenue;
- tax distribution median, p95, p99, and maximum;
- reward concentration by day and difficulty interval;
- reorg observations around unusually high-tax blocks;
- miner/template implementation diversity.

Node operation

- expiry-index size and growth;
- commitment mismatch count;
- index rebuild time;
- pruned and archive-node storage;
- snapshot import and verification time;
- archive-node count and distribution.

---

18. Legal and compliance notice

OBTC is an independent chain and is not the same as Bitcoin.
The expiry and reclaim mechanism may be interpreted in different jurisdictions as a protocol fee, a negative interest mechanism, or a redistribution rule applied to dormant assets. Exchanges and custodians should design their internal processes and user disclosures accordingly.

---

19. Glossary

Term	Meaning	Role in OBTC
UTXO	An unspent transaction output	The lifecycle accounting unit
OutPoint	`txid:vout`, the unique locator of an output	Identifies each UTXO deterministically
Creation height	The block height where a UTXO was created	Starting point for expiry calculation
Expiry height	`creation height + WindowBlocks`	Height where ordinary spending closes
Expired UTXO	A live UTXO at or beyond expiry height	Eligible for REAP and invalid for ordinary spend
Pending expired	Expired but not yet included in REAP	Remains in the canonical queue
Renewal	A holder-signed spend before expiry	Resets lifecycle through new outputs
REAP	Reclaim Expired Assets Protocol	Miner-constructed block-internal system transaction
Refund	The retained value recreated under the original script	Preserves a direct ownership path after normal REAP
Tax	Protocol-defined value credited to miner reward	Contributes to PoW security budget
Dust fold	Full-tax treatment for expired input below `720 sat`	Ends the lineage without a refund output
Canonical order	`expiry -> amount -> outpoint`	Removes miner discretion over input choice
Canonical prefix	The first `k` eligible entries under canonical order	Must exactly match included REAP inputs

Term	Meaning	Role in OBTC
Cohort-local amount order	Amount ordering only among equal-expiry UTXOs	Creates small-to-large progression inside one expiry cohort
REAP marker	`REAP:height:count:digest` OP_RETURN payload	Binds block height and ordered inputs
Normal input cap	`256` mainnet inputs	Limits refund-producing REAP work
Dust input cap	`1,024` mainnet inputs	Allows higher refundless cleanup throughput
REAP max weight	Current mainnet consensus limit of `400,000 WU`	Caps any single REAP transaction; the default template budget follows this value
Expiry index	Ordered state for creation/expiry tracking	Supports candidate discovery and prefix validation
Expiry commitment	Coinbase commitment to indexed expiry state	Detects state divergence across validators
Mempool	Pool of ordinary unconfirmed transactions	REAP is not accepted as a user-relayed transaction
Coinbase	Block reward settlement transaction	Claims subsidy, fees, and permitted REAP tax
Replay protection	OBTC-specific signature-domain enforcement	Prevents cross-chain signature reuse
Namespace isolation	Distinct network, address, key, and port identifiers	Reduces accidental cross-chain confusion
Pruned full node	Validating node without all historical block bodies	Reduces local storage requirement
Archive node	Node retaining complete historical blocks	Supports audit and full-history query
Reorg	Active-chain replacement near the tip	Requires exact lifecycle-state rollback

Term	Meaning	Role in OBTC
Liveness	Continued chain or queue progress	Chain liveness is guaranteed independently of REAP inclusion; REAP queue speed is not

20. Frequently asked questions

Q1: What happens if I do nothing?

When your UTXO reaches its expiry height, it can no longer be spent through the ordinary path. It enters the canonical expired queue and may later be processed by REAP.
For a normal input, REAP creates a 70% refund under the original script and credits 30% as miner tax.

Q2: Does expiry mean an immediate 30% deduction?

No. Expiry changes the UTXO's protocol state. Tax and refund are created only when a miner includes the corresponding canonical REAP prefix in a block.
Until then, the UTXO is pending expired: it cannot be spent normally, but it has not yet produced a refund.

Q3: How do I avoid REAP?

Spend or renew before expiry. Wallets can expose expiry height, warning windows, batch dry-runs, manual renewal, and optional automation.

Q4: Why is the tax 30%?

The 30% rate leaves a substantial 70% refund while making permanently dormant value decay at a meaningful but still gradual rate.
At a 6.904-year lifecycle:

- value halves approximately every 1.943 lifecycles, or about 13.4 years;
- an isolated 1-OBTC lineage takes 34 normal cycles to produce a refund below 720 sat;
- it is fully folded on the 35th event, approximately 241.6 years under immediate-processing assumptions.

A 20% rate would extend the same example to about 379.7 years; a 40% rate would shorten it to about 172.6 years. The selected rate is therefore a middle path between user refund continuity and finite dormant-value extinction.

Q5: Why is the mainnet window 362,880 blocks?

`362,880 = 9! = 180 x 2,016`. It is approximately 6.904 years at the nominal ten-minute target and is often described as about seven years.

The factorial identity explains the constant's provenance; the practical rationale is a long holder-maintenance interval aligned to an integer number of standard 2,016-block periods.

Q6: Why is there a 720-sat dust threshold?

`720 = 6!`. More importantly, a fixed threshold gives the protocol a deterministic point where creating another refund output is no longer worthwhile.
Inputs below 720 sat are fully folded into tax. The protocol does not depend on a fee oracle or governance-updated dust value.

Q7: Does every REAP transaction reduce the UTXO set?

No.

- normal REAP usually consumes one UTXO and creates one refund UTXO;
- dust-fold REAP consumes one UTXO and creates no refund;
- user renewal can reduce UTXO count when it consolidates several inputs.

REAP guarantees value metabolism, not immediate one-for-one state reduction at every normal event.

Q8: Why are inputs ordered by expiry, amount, and outpoint?

Expiry order preserves oldest-first processing. Amount order applies only inside the same expiry cohort and creates a deterministic small-to-large progression. Outpoint provides a unique tie-break.
The miner cannot skip ahead to a larger output. Every validating node derives and checks the same prefix.

Q9: Must every block include REAP?

No. REAP inclusion is optional at consensus, while REAP validity is strict when present.
This prevents a local planner failure from stopping block production, but it also means the protocol does not promise a fixed processing date for each expired UTXO.

Q10: Why does REAP not enter the mempool?

REAP is a system transaction assembled from consensus-visible expired state. Relaying user-created lookalike transactions would add spam, race conditions, and ambiguity without giving users control over canonical selection.

Q11: Are replay protection and namespace isolation redundant?

No.

- namespace isolation reduces address, key, port, and operator confusion;
- replay-protected signature domains prevent cross-chain signature reuse.

Q12: Can a miner select only the most valuable expired outputs?

No. If the miner includes REAP, the inputs must be the exact canonical global prefix. Skipping, substituting, or reordering inputs invalidates the block.

Q13: Is 400,000 WU only a policy target?

No. On mainnet, `400,000 WU` is the current consensus `ReapMaxWeight`. Mining builders also use it as the default REAP template budget, but validators reject a REAP transaction above this limit. The input caps, tax rule, dust rule, canonical order, and ordinary block limits remain independently enforced.

Q14: Why is the historical backlog large at launch?

The chain inherits old UTXO creation heights. When expiry activates, historical outputs older than the lifecycle boundary are immediately eligible.
Input caps and block weight prevent a one-block purge, so the inherited set becomes a multi-year ordered queue under extreme no-renew assumptions.

Q15: Why is this not confiscation?

The rule is precommitted, uniform, and avoidable through renewal before expiry. After a normal REAP event, most value returns directly to the original script. No operator or committee chooses the owner, recipient, timing order, or tax share.
The design still represents a different ownership contract from Bitcoin. Its legitimacy depends on clear rules, long notice, reliable wallet tooling, and users understanding the lifecycle before relying on the chain.

Q16: Which sections matter most to implementers?

Recommended order:
`Section 5 -> Section 7 -> Section 8 -> Section 9 -> Section 11 -> Section 13 -> Section 14`

---

21. Conclusion

OBTC is a Bitcoin-derived monetary system built around an explicit UTXO lifecycle.
Its core rule set is compact:

- a mainnet UTXO expires after `362,880` blocks;
- active holders can renew before expiry;
- expired outputs leave the ordinary signing path;
- miners may process them through a canonical REAP transaction;
- normal inputs produce a 70% refund and 30% miner tax;
- inputs below `720 sat` terminate without another refund;

- input order, tier ceilings, coinbase accounting, replay protection, and expiry commitments constrain the system path;
- wallet and mining policy remain separate from monetary consensus wherever possible.

The design does not claim that dormancy proves loss, that REAP instantly shrinks all state, or that miner revenue is guaranteed. It makes a different claim:

> A long-term ownership system can preserve active claims while giving permanently inactive value a deterministic, gradual, and ultimately finite lifecycle.

The success criterion for this whitepaper is implementation convergence:

> Can independent teams implement the rules and accept exactly the same valid chain behavior?

The success criterion for the network is operational evidence:

> Can holders renew safely, can miners process expired state without excessive block pressure, and can nodes keep lifecycle state consistent through normal operation and reorganization?

---

22. License

End of Organic Bitcoin Whitepaper V1.1 (Implementation-Aligned, Extended Readable Edition)