DUCAT PROTOCOL
AUDIT REMEDIATION

Prepared for DUCAT Protocol

Review date 27 October 2026 · closeout verified 5 November 2026

Heads reviewed
protocol-sdk #120 at `872fa40` · 20 Oct 2026, closeout `69f0e54`
validator-rs #101 at `379ff54` · 20 Oct 2026, closeout `dedd1d4`
guardian-rs #101 at `9f0e566` · 20 Oct 2026, closeout `6e76f02` / `80b2e47`
core-ts `d9fce5c` · ducat-infra closeout record `ducat-2054-final-audit-closeout-2026-11`

Contents

1. About SetDev 3

2. Introduction 4

3. About DUCAT Protocol 5

   3.1 Protocol Components 6

   3.2 System Graph 6

   3.3 Price Action System 7

   3.4 Vault Action System 7

   3.5 Validator Pipeline 8

4. Risk Classification 6

5. Remediation Executive Summary 7

   5.1 Protocol Summary 7

   5.2 How the Remediation Moved 7

   5.3 Summary of Findings 8

6. Findings 9

   6.1 Critical Findings 9

   6.2 High Findings 9

   6.3 Recommended Findings 11

   6.4 Informational / Optimization Findings 17

7. Residuals from the 20–26 October Re-read 27

8. Other Tickets on These Heads 29

9. Conclusion 30

P2-01 The FROST pin inverted `tweak(None)` relative to upstream **Closed**

P2-02 Unvalidated price-bucket terms could skip the liquidation check **Closed**

P2-03 Float-to-integer conversions failed open **Closed**

P2-05 Guardian-to-validator traffic is unauthenticated HTTP **Accepted**

P2-06	`UnitAmount` addition wrapped on consensus balance checks	Closed
P2-07	`UnitBalanceMin` was not a shared predicate	Closed*
P2-08	The client SDK did not bind guardian responses to the request	Closed*
P2-09	`get_threshold_price` did not share TypeScript's post-conditions	Closed
P2-10	Decimal scale came from a language-specific printer (was P1-02)	Closed
P3-01	TypeScript does not check the committed threshold against the vault	Accepted
P3-02	Oracle commit age is not enforced in the TypeScript guardian	Closed*
P3-03	The TypeScript guardian does not bind liquidation reserve payouts	Closed*
P3-04	The strict borrow verifier omits the collateral-ratio floor	Closed*
P3-05	Limit helpers & three policy flags are unused	Closed*
P3-06	Vault script parsers accept non-curve public keys	Closed*
P3-07	Verifier exports do not show which checks are complete	Closed*
P3-08	`hasher_to_scalar` could panic & disagreed with upstream	Closed
P3-09	Participant final-signature verification was unreachable	Closed
P3-10	The coordinator accepted shares from outside the signing package	Closed
P3-11	The coordinator panicked on its own share if the registry was disabled	Closed
P3-12	The libp2p identity is derived from the FROST signing share	Accepted
P3-13	FROST nonces were not zeroized on failure paths	Closed
P3-14	Huffman taproot builders were public	Closed
P3-15	The breached-commit filter existed only on the Rust tree builder	Closed*
P3-16	`TAPLEAF_VERSION` is `0x51` & unused	Closed*
P3-17	Merkle parity is pinned in one direction; fixture generator missing	Closed*
P3-18	libp2p identity binding is optional without a peer registry	Closed
P3-19	Validator statistics wrapped on raw addition	Closed
P3-20	The validator HTTP API is unauthenticated	Closed

* Closed outside the three pull requests reviewed here: P2-08 in client-sdk / ducat-infra #70; P3-02 in ducat-infra #92 & core-ts #27; P3-04, P3-05, P3-06, P3-07 in core-ts (#27, `d9fce5c`); P3-15, P3-16, P3-17 in client-sdk #84 (& validator-ts #36 for P3-17's pin); P2-07 & P3-03 by guardian-ts test evidence in the November closeout record.

1. About SetDev

SetDev is a product & research firm focused on Bitcoin, Bitcoin L2s, & the smart contract ecosystems that connect to them. We partner with protocols & companies across the Bitcoin stack—from low-level primitives to emerging app layers—to ship software, publish technical research, & contribute to open-source development.

We've collaborated with some of the most forward-thinking teams in the Stacks & Bitcoin ecosystem. Our audits are conducted by engineers with hands-on experience building & shipping production-grade systems in Rust, TypeScript, & Clarity.

Learn more about SetDev at setdev.com

2. Introduction

This document presents the results of a remediation review conducted by SetDev for DUCAT Protocol. The original audit was performed by Jesus Najera (Rust repositories) & Bernardo Garciarivas (TypeScript repositories) between June & August 2026. It re-examines the 29 protocol findings of the August 2026 security audit against the three October consolidation pull requests: protocol-sdk #120, validator-rs #101, & guardian-rs #101.

Alongside manual review, the audit & this remediation pass used several language models as reading & cross-checking aids: Qwen, Kimi, DeepSeek, Claude, & OpenAI models. Model output was treated as a lead, never as a finding; every claim in this report was confirmed against the source on the referenced commits.

For every August finding this report records the original issue, the concern behind it, where the code stands on the reviewed heads, how the remediation travelled through the September ticket pull requests into the October consolidation, & what, if anything, remains before the finding can be closed.

Line numbers cited below are on the three pull-request heads, not on dirty working trees. The ducat-infra issue tracker is private & was not readable from this session, so each section is keyed by its August finding ID. P3-21, a gitignored `.npmrc` on a local working copy, is not a protocol defect & is omitted.

On 5 November 2026 DUCAT delivered a closeout record (`ducat-2054-final-audit-closeout-2026-11`, ducat-infra) with new commits on the same DUCAT-2054 pull requests plus evidence & replies for the remaining items. SetDev verified the linked commits & tests. Each affected finding below carries a *November closeout* note; three items DUCAT chose to keep as designed are marked **Accepted**.

A final pass on 20–26 October re-read the commits that landed after the first comment batch. Several proposed comments did not survive that re-read & are recorded as such; the new review items that did survive are collected in Section 7.

Scope & method

Five of the protocol's seven repositories were in scope for the original audit: protocol-sdk, guardian-rs, validator-rs, core-ts, & client-sdk. ducat-infra (development environment & test stack) & bootstrap-ts (anchor bootstrap) were helpers outside the audit boundary. DUCAT has no virtual machine & therefore no re-entrancy; the audit concentrated on transaction construction, the two-transaction move & update process, the FROST threshold-signature scheme, & cross-language drift between the Rust & TypeScript implementations.

The review ran top-down first — a catastrophe analysis of the interfaces most likely to move funds, a line-by-line rewrite of those sites, an invariant list, & fuzz tests against it — then bottom-up through each repository. The core question throughout was whether any path can move, hide, or destroy collateral outside the two intended outcomes (repay or redeem, & liquidate below threshold), or misclassify a rule as strict versus policy.

This remediation review applies the same lens to the October pull requests: for each August finding it asks whether the fix landed on the consensus path, whether it landed on both sides of the language boundary, & whether anything new was introduced alongside it.

3. About DUCAT Protocol

DUCAT is a Bitcoin-backed loan & stablecoin protocol built on Runes & Taproot primitives. Users lock BTC collateral in taproot vaults & borrow UNIT, the protocol's stablecoin, against it (USDC is also available). Liquidation thresholds are committed as price buckets in the vault's taproot tree & are authorized by signed oracle price contracts, so a vault becomes liquidatable when the oracle publishes a price at or below its committed leaf.

There are two kinds of participants. **Guardians** are empowered operators that co-sign critical operations through a FROST threshold-signature federation. **Clients** are end-users that borrow, lend, or liquidate. Because there is no virtual machine, protocol rules are enforced by transaction validation on both sides of a signing request rather than by on-chain execution.

The protocol is split across seven repositories. Five were in scope for the original audit; the three Rust heads are the subject of this remediation review, with the TypeScript pull requests verified for the TS-side findings:

REPOSITORY	LANGUAGE	ROLE	AUDIT SCOPE	THIS REVIEW
protocol-sdk	Rust	Protocol rules & transaction validation	In scope	**#120**
guardian-rs	Rust	Co-signing & FROST MPC	In scope	**#101**
validator-rs	Rust	Chain indexer & state engine	In scope	**#101**
core-ts	TypeScript	Protocol reference implementation	In scope	**#27**
client-sdk	TypeScript	Client & wallet integration	In scope	**#84**
ducat-infra	TypeScript	Development environment & test stack	Helper	**#92**
validator-ts	TypeScript	Chain indexer (TypeScript)	—	**#36**
bootstrap-ts	TypeScript	Anchor bootstrap	Helper	—

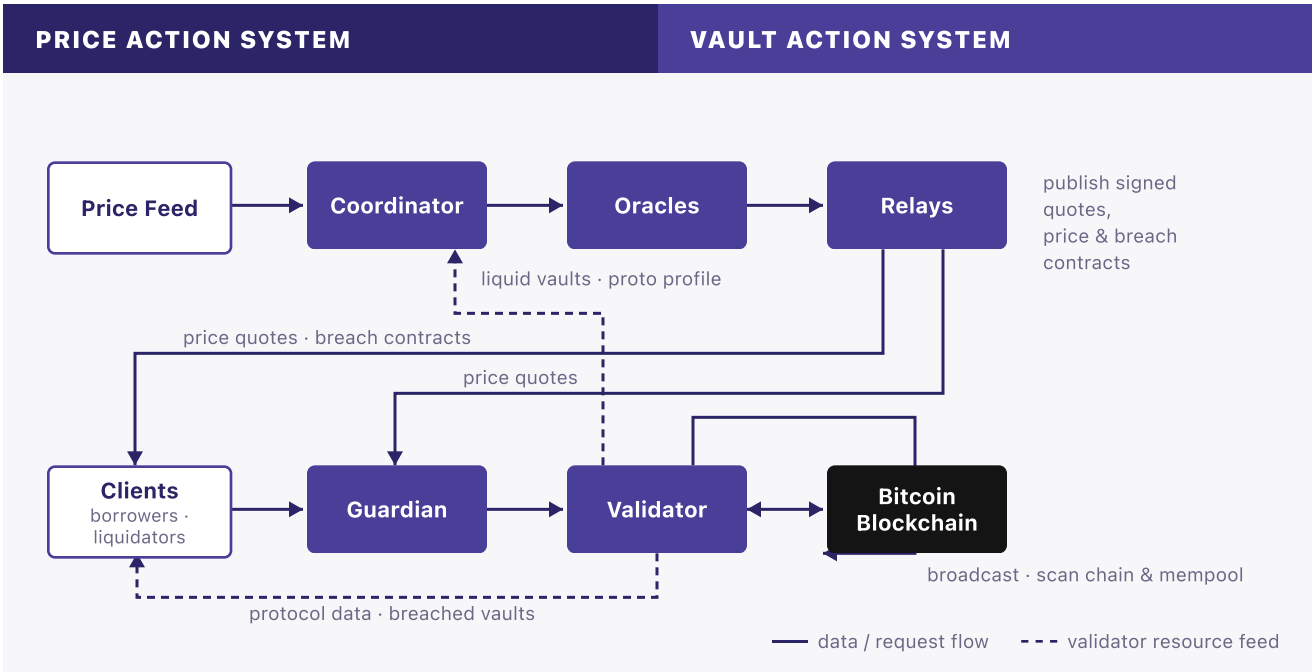
Because liquidation leaves, bucket arithmetic, & taproot roots are consensus-critical, the recurring

theme of the August audit was *parity*: both stacks must compute the same leaf, the same scale, & the same tree. Much of this remediation review is therefore about whether a fix landed on the Rust side only, or on both.

3.1 Protocol Components

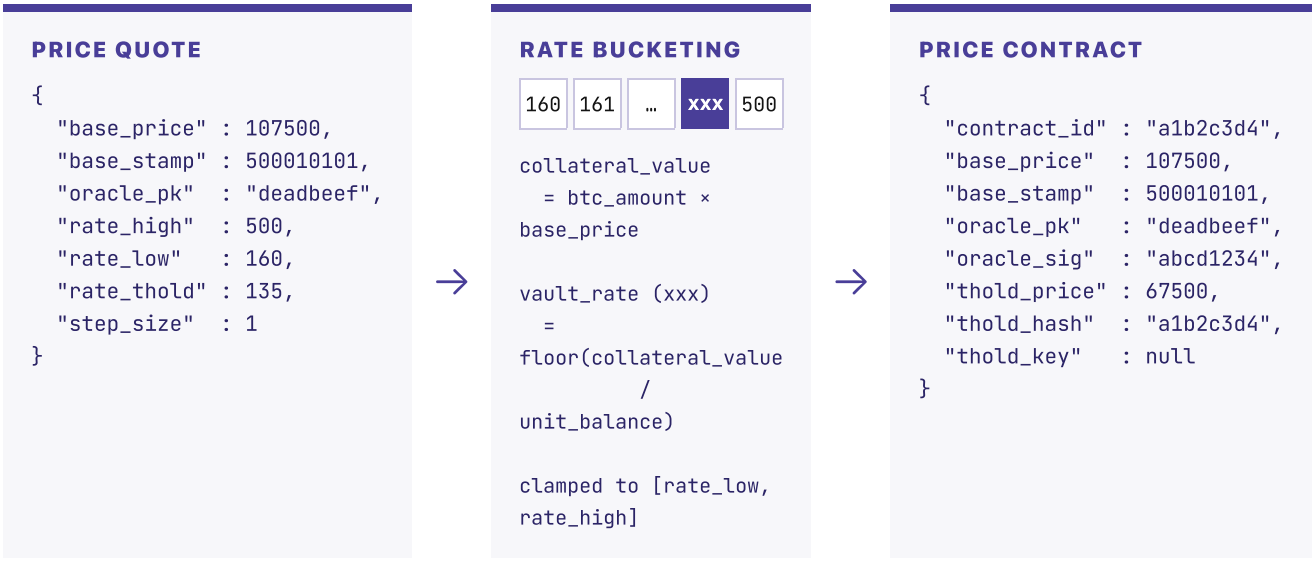
COMPONENT	ROLE	CONSUMES	PRODUCES
Coordinator	Fetches the exchange price, builds price quotes, packages requests for the oracles.	Price feed, liquid vaults & proto profile (validator), custom price terms	Price quotes, request packets
Oracle server	Verifies each request, generates the rate bucket & breach contract keys, signs the contracts.	Request packets, price feed	Signed price contracts
Relay	Publishes signed oracle events to subscribers.	Signed events	Price quotes, price & breach contracts
Guardian node	Receives vault requests, validates, FROST-signs, finalizes & broadcasts them.	Price quotes (relay); asset accounts, liquid vaults, vault & proto profile (validator)	Signed, broadcast vault transactions returned to the client
Validator node	Scans chain & mempool for protocol actions; serves assets, records, vaults & breached vaults.	Bitcoin blockchain, node mempool	Proto profile, vault profile, asset accounts, active & liquid vaults, pending actions
Borrower client	Subscribes to quotes, configures a vault action, selects inputs, approves signing.	Price quotes, validator data	Vault action requests
Liquidator client	Subscribes to breach contracts, fetches breached vaults, completes liquidation actions.	Breach contracts, breached vaults	Liquidation requests

3.2 System Graph


PRICE ACTION SYSTEM
VAULT ACTION SYSTEM
Price Feed
Coordinator
Oracles
Relays
publish signed quotes, price & breach contracts
liquid vaults · proto profile
price quotes · breach contracts
price quotes
Clients
borrowers · liquidators
Guardian
Validator
Bitcoin Blockchain
broadcast · scan chain & mempool
protocol data · breached vaults
data / request flow
validator resource feed

3.3 Price Action System

The coordinator turns a spot price into a **price quote**: a base price, a stamp, & a rate grid. Oracles sign one **price contract** per rate bucket. A borrower's vault selects the bucket that matches its collateral ratio, & the contract's `thold_price` becomes the liquidation leaf committed in the vault's taproot tree. P2-10 (was P1-02), P2-02 & P2-09 all live in this pipeline.



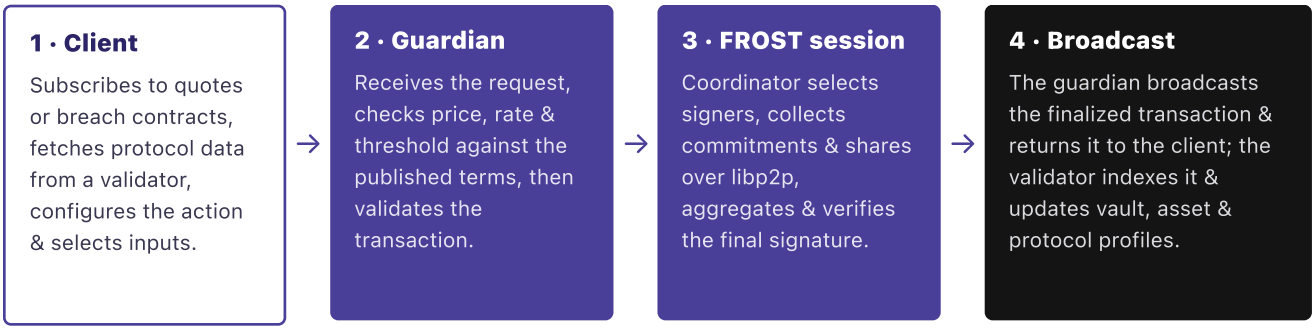
Built by the **coordinator** from the price feed; delivered to oracles & relays.

Computed by the **oracle** for each step between `rate_low` & `rate_high`.

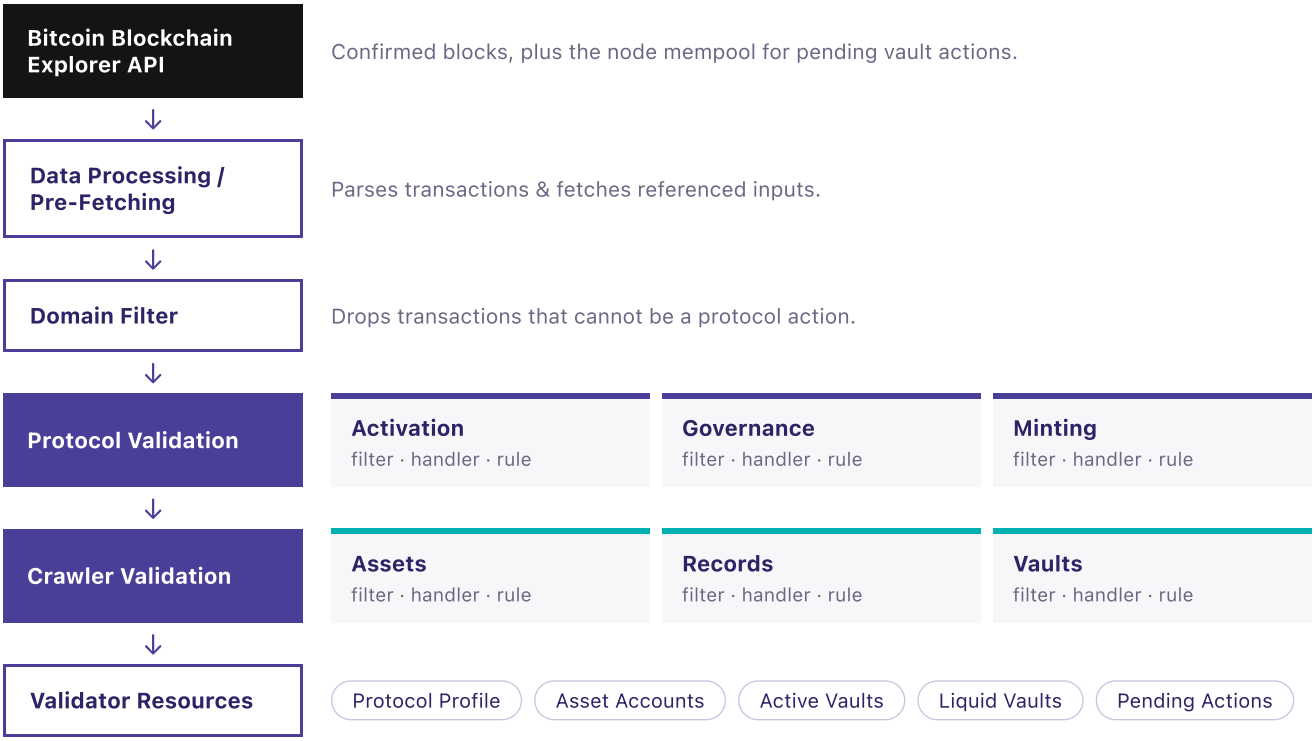
One signed contract per bucket, plus breach-contract keys & custom contracts; delivered to relays.

3.4 Vault Action System

A borrower or liquidator builds a vault action against the current quote & submits it to a guardian. The guardian validates the request (locally or through a validator), runs a FROST signing session with its peers, & broadcasts the result. P2-03, P2-05 & the session findings P3-09 through P3-13 sit on this path.



3.5 Validator Pipeline



Served to clients, guardians, & the coordinator (breached vaults). P3-19 & P3-20 concern the statistics & the public API at the bottom of this pipeline; the residuals in Section 7 concern the vault routes & block-abort rules inside it.

4. Risk Classification

Each finding in this report is assigned a priority level (P0–P3) to indicate the relative severity of the issue & the urgency of addressing it. The engagement scope framed the same levels in protocol terms: **fund loss** (P0) — any path that moves collateral outside the intended outcomes; **state corruption** (P1) — a corrupted or diverged state that requires a change to some part of the stack, including cross-language type & format drift; & **state disruption** (P2) — a temporary suspension of service that needs no change to an immutable part of the stack. P3 covers hygiene & best practice.

P0 – Critical	These issues have a high certainty of resulting in loss of protocol or user funds. They are the most severe & require immediate remediation.
P1 – High	These issues represent vulnerabilities that are less likely to be exploited but could lead to significant harm under certain conditions, such as admin error or governance misuse. They should be addressed prior to deployment or as soon as possible.
P2 – Recommended	These findings are unlikely to cause direct loss but may lead to unexpected behavior, degraded user experience, or unnecessary risk exposure. Fixing them is strongly encouraged for robustness.
P3 – Informational	These items are optional improvements, typically involving clarity, hygiene, or best practices. Addressing them is recommended to improve code quality but is not required for security.

Remediation status labels

CLOSED	The finding as written is resolved on the reviewed heads.
PARTIAL	One half of the recommendation landed, typically the Rust side; the other half is outside these pull requests.
OPEN	The finding is unchanged, or the change that shipped does not close it.
ACCEPTED	DUCAT has chosen not to change the behaviour, documented the reasoning, & SetDev has accepted the design choice. The original concern is recorded for the record.

5. Remediation Executive Summary

This review assessed how the August 2026 findings were remediated across the October consolidation of protocol-sdk, validator-rs, & guardian-rs. Of the 29 findings, after the November closeout, **15 are closed** on these heads, **11 are closed** in TypeScript pull requests outside these heads, & **3 are accepted** design choices. None remain open or partial.

> **Assessment:** The numeric & signing-mode remediation is sound & is on the reviewed heads. No high-severity findings remain. The November closeout resolved every remaining open or partial item: new commits for P3-04 / P3-07, R-01 & R-07, test evidence for P2-07, P3-03 & P3-18, API changes for R-02 / R-03, & documented design decisions for P2-05, P3-01 & P3-12, which SetDev accepts. The seven Section 7 residuals are all resolved.

5.1 Protocol Summary

Protocol name	DUCAT Protocol
Repositories	DUCAT-UNIT/protocol-sdk · DUCAT-UNIT/validator-rs · DUCAT-UNIT/guardian-rs
Review date	27 October 2026 · closeout verified 5 November 2026
Protocol type	Bitcoin-native collateralized vault / FROST federation signer
Languages	Rust (reviewed heads) · TypeScript (parity counterpart, unchanged)
Original auditors	Jesus Najera (Rust) · Bernardo Garciarivas (TypeScript)

5.2 How the Remediation Moved

The August report named 29 protocol findings: nine recommended & twenty informational, after the two findings originally filed as high were downgraded to P2 on 26 August. Through September those were filed as separate DUCAT tickets & reviewed as small ducat-infra pull requests: #77 for decimal scale (DUCAT-1973), #80 for checked numeric conversion (DUCAT-1976), #74 for checked addition (DUCAT-1977), #79 for threshold post-conditions (DUCAT-1980), & #70 for client response binding (DUCAT-1979). All five were accepted as closing their findings, with nits that did not block.

October folded that work into three large pull requests. protocol-sdk #120 carries the consensus rules. validator-rs #101 & guardian-rs #101 consume those rules & add the service-side tickets (HTTP limits, statistics bounds, signing-session hygiene, deployment policy, presigning).

A last pass on 20–26 October re-read the commits that landed after the first comment batch. Several proposed comments did not survive that re-read. The presign stack check is one of them: the guardian inserts the missing signature placeholder before it calls authorize, & a bad witness is rejected. The ignored mint test shows a validation gap; it does not, by itself, install a copied etching as the reserve.

5.3 Summary of Findings

SEVERITY	COUNT	CLOSED	ACCEPTED	OPEN	STATUS
P0 – Critical	0	–	–	–	N/A
P1 – High	0	–	–	–	Two August P1s downgraded to P2 (26 Aug)
P2 – Recommended	9	8	1	0	P2-08 closed in client-sdk; P2-05 accepted (embedded validator on mainnet); P2-10 was P1-02
P3 – Informational	20	18	2	0	P3-01 (quote bucket check stays client-side) & P3-12 (share-derived peer identity) accepted
Total	**29**	**26**	**3**	**0**	11 of the 26 closures are outside the three reviewed PRs: P2-07, P2-08, P3-02, P3-03, P3-04, P3-05, P3-06, P3-07, P3-15, P3-16, P3-17.

6. Findings

6.1 Critical Findings

No critical (P0) findings were identified in the August audit, & none were introduced by the reviewed pull requests.

6.2 High Findings

No high (P1) findings remain. The two issues filed as P1 in August (`rate_max` snapping & decimal-scale counting) were downgraded to P2 on 26 August in agreement with DUCAT; the decimal-scale finding is carried below as P2-10. The `rate_max` finding was withdrawn from this report.

6.3 Recommended Findings

[P2-01] The FROST pin inverted `tweak(None)` relative to upstream

CLOSED

Location: guardian-rs `Cargo.toml` (pin `conduition/frost@59d1da2`); protocol-sdk `crates/ducat-protocol/src/signable.rs:640-646` · DUCAT-1974

ORIGINAL FINDING

The pinned fork treats `tweak(None)` as no tweak. Upstream `frost-secp256k1-tr` 2.1.0 & later treats `None` as the BIP-341 hash-only tweak. Call sites were internally consistent (script-path `None`, key-path `Some([])`), but a routine bump would sign vault leaves with a tweaked key while Bitcoin script checks the raw leaf key. Every guardian-signed vault spend would fail.

CONCERN

This is a halt, not a theft. Funds that require a federation signature stop moving. The types do not change, so the compiler does not warn.

WHERE THE CODE IS NOW

`SigningTweak` is now an enum: `ScriptPath`, `KeyPathNoTree`, or `KeyPathWithTree([u8; 32])`. The comment at lines 630–638 says serialization keeps the legacy wire form on purpose: null is script-path, an empty array is key-path with no tree, 32 bytes is key-path with a tree. Translation to the fork's `Option` happens in that one impl. guardian-rs #101 `d43ab38` (18 October) integrates FROST 3 with those checked tweaks; the vendored crate is no longer the pre-merge snapshot the audit read.

HISTORY

DUCAT-1974 is protocol-sdk `2b80eaf` (18 October), "consolidate explicit FROST 3 signing modes." guardian-rs took `d43ab38` the same day. The encapsulation arrived with the FROST 3 consolidation, not with the September numeric PRs.

WHAT REMAINS

An upstream bump is still consensus-sensitive, & the comment on the enum says so. The silent `None` inversion the audit described is no longer the type callers use.

[P2-02] Unvalidated price-bucket terms could skip the liquidation check

CLOSED

Location: protocol-sdk `price_bucket.rs:29-70` (`parse_bucket_terms`), `:106-114` (`evaluate_bucket`); oracle `relay.rs:151`

ORIGINAL FINDING

`AnchorTermEntry::validate` skipped `PriceBucketMin`, `PriceBucketMax`, & `PriceBucketSize`, & `check_liquidation_price` returned `Ok(())` when `step_scaled == 0` without comparing `thold_price`. A zero, subnormal, or NaN step on a governance anchor would unpin liquidation thresholds for every vault under that anchor.

CONCERN

One bad anchor term disables the price check for the whole generation of vaults created under it. The failure is silent success, which is the wrong direction for a liquidation gate.

WHERE THE CODE IS NOW

`parse_bucket_terms` rejects a non-decimal, non-positive, or unordered term (`invalid_decimal`, `non_positive`, `invalid_range`, `invalid_threshold`) before any bucket is computed. A step not on scale 1, 10, or 100 is `unsupported_step`. `evaluate_bucket` returns `invalid_input` when the step or threshold is zero, when debt or base price is zero, or when divisibility exceeds 38. There is no success path for a zero step. The oracle relay calls `parse_bucket_terms` before `evaluate_bucket`.

HISTORY

A September review accepted this as closed: validation on decode, `check_liquidation_price` erroring on a zero step, & the same checks in core-ts. The nit was that scale still came from the printer, which was P2-10. The October head keeps the checks & replaces the printer. DUCAT-2053 (`ba9be0b`) is the commit that put this parser on the shared foundation.

WHAT REMAINS

core-ts is outside these diffs. If a TypeScript anchor decoder still skips the three bucket terms, that half is still open there. The Rust indexing path no longer has the `Ok(())` bypass.

[P2-03] Float-to-integer conversions failed open

CLOSED

Location: protocol-sdk `calc.rs:50` (`checked_f64_to_u64`), `amount.rs`; guardian signing gate · DUCAT-1976 · closed on all three heads

ORIGINAL FINDING

`ToFixedPrice` for `f64` used `floor() as u32`. `max_sats_withdrawable` returned `Amount::ZERO` on conversion failure. `UnitAmount` & `DucatAmount` multiplication by `f64` truncated & saturated. The guardian signing gate parsed the remote liquidation threshold with `as_f64()` & no finite or range check, so NaN made the under-collateralization comparison false. `checked_f64_to_u64` already existed & was unused at these sites.

CONCERN

A NaN or overflow becomes a real protocol amount: zero, which looks like "nothing to withdraw" or "not undercollateralized," or `u32::MAX` / `u64::MAX`, which looks like an enormous price or balance. Either accepts or sizes a spend the formula did not compute.

WHERE THE CODE IS NOW

`checked_f64_to_u64` is the conversion `UnitAmount` & `DucatAmount` call. Tests in `calc.rs:289-305` & `tests/numeric_conversions.rs` reject NaN, infinities, negatives, & values past the exact integer range. `Mul<f64>` as a saturating operator is not the production impl on this head; the methods return `Result`.

HISTORY

ducat-infra #80 was reviewed on 30 September & accepted: `ToFixedPrice` as `checked_nonnegative` plus `checked_f64_to_u32`, `max_sats_withdrawable` returning `Err`, `checked_mul_float` / `checked_div_float`, & the guardian ingress running `LiquidationThold.validate()` over `[MIN_LIQUIDATE_THOLD, MAX_LIQUIDATE_THOLD]`. October consolidated that stack: protocol-sdk `c8d20e3`, validator-rs `2f498f6`, guardian-rs `0d18056` (18 October).

WHAT REMAINS

The parity harness is the regression gate from the September review (`make test-numeric-parity`). These heads do not add a new float bridge at the sites the audit named.

[P2-05] Guardian-to-validator traffic is unauthenticated HTTP

ACCEPTED

Location: guardian-rs `guardian-core/src/validator_backend/remote.rs`, `guardian-exe/src/config.rs`; compose `GUARDIAN_EXTERNAL_VALIDATOR_URL` · DUCAT-1982

ORIGINAL FINDING

Validation verdicts, guardian identifiers, & the liquidation threshold were fetched over cleartext HTTP, & every guardian pointed at one validator. A compromised or intercepted validator could present the same forged verdict to the whole federation. The recommendation, past local regtest, was the embedded validator, localhost, or mTLS, & to stop treating shared HTTP queries to one validator as independent attestation.

CONCERN

This is the trust boundary under P2-03's signing gate. If the remote verdict is the only check, forging it forges the gate.

WHERE THE CODE IS NOW

`378d6bf` (15 October, DUCAT-1982), "require independent Guardian validation outside local regtest," touches `remote.rs`, `config.rs`, & new deployment-policy tests under `guardian-debug`. Non-regtest guardians are not allowed to treat the remote validator as sufficient. The HTTP client itself is still HTTP; this commit does not wrap it in mTLS.

HISTORY

An early look at guardian-rs #101, when the branch was described as DUCAT-2054 only, said not to close 1982 on the bootstrap work. 1982 landed later as its own commit & is on `9f0e566`. ducat-infra #73 was the earlier home of this ticket.

WHAT REMAINS

The deployment rule matches "do not trust one remote validator outside regtest." The channel is still cleartext wherever a deployment sets `GUARDIAN_EXTERNAL_VALIDATOR_URL`. Leave the finding open until that channel is authenticated, or narrow the ticket to the deployment policy & close it with that scope written down.

NOVEMBER CLOSEOUT

DUCAT confirmed the deployment rule: production & all mainnet configurations require the embedded validator, which the guardian calls in-process with no HTTP. Remote validation is restricted to explicitly enabled local regtest. This is the "narrow the ticket to the deployment policy & write it down" option; SetDev accepts it. The cleartext channel exists only in a configuration that cannot reach mainnet.

[P2-06] `UnitAmount` addition wrapped on consensus balance checks CLOSED

Location: protocol-sdk `asset/unit/amount.rs:246, 251`; `borrow/validated.rs:675, 882` · DUCAT-1977 · closed on all three heads

ORIGINAL FINDING

Addition wrapped in release & panicked in debug; subtraction saturated to zero. A wrapping add at the borrow balance-equality check could, in principle, make `prev + issued` equal the new balance after an overflow. Reaching that needs balances near `u64::MAX` cents. The recommendation was checked addition & a validation error at the consensus sites.

CONCERN

The economic precondition is absurd without another bug, which is why this was P2 & not P1. The shape is still wrong: a consensus equality check that wraps can be satisfied by an overflow instead of by the amounts the user signed.

WHERE THE CODE IS NOW

`UnitAmount::checked_add` & `try_sum` exist, & the type comment says to use them for untrusted balances. Borrow validation defines `DebtAmountOverflow` & uses it; tests expect that error.

HISTORY

ducat-infra #74 was reviewed on 30 September & accepted (protocol-sdk `a20acf3`, `8a298b8`, `df25619`). `+` now panics in debug & release, so wrapping is gone even without workspace `overflow-checks`; core-ts `checked_add_amount` matches. October consolidated the fix as protocol-sdk `6f2ea53`, validator-rs `69c8f8b`, guardian-rs `53f8fdc`, all "reject overflowing asset & debt sums."

WHAT REMAINS

Saturating subtraction is still on the amount type if a caller uses `-`. That was out of scope in September & remains so. The wrapping add the finding described is gone.

[P2-07] `UnitBalanceMin` was not a shared predicate

CLOSED

Location: protocol-sdk `repay/validated.rs:171, 370`, `trim/validated.rs:163-168, 368, 449`; core-ts, guardian-ts, validator-ts evaluators · DUCAT-1978

ORIGINAL FINDING

`UnitBalanceMin` requires debt to be zero or at least the configured minimum. The client SDK rejected a partial repay that would leave dust; the TypeScript guardian & validator did not. Rust rejected open & borrow increments below the term, & did not check remaining debt after a partial repay. The recommendation was to apply the predicate on open, borrow, repay, & trim in the TypeScript evaluators, & to mirror the remaining-balance check in protocol-sdk repay & trim.

CONCERN

Dust debt is a position the liquidation & repay math was not written for. A vault that looks closed in the UI & still carries a few cents of debt stays in the liquidatable set.

WHERE THE CODE IS NOW

protocol-sdk #120 defines `DebtBelowMinimum` on repay, & `LiquidatorDebtBelowMinimum` / `TargetDebtBelowMinimum` on trim, applied on both sides of a trim. Tests `minimum_debt_boundaries_validate_raw` & `minimum_debt_boundaries_both_trim_vaults` cover them. The TypeScript half has since landed outside these pull requests: core-ts #27 (`d85059f6`, 16 October) enforces the predicate as a **strict rejection** — resulting debt must be zero or at least `unit_balance_min` — across open, borrow, repay, withdraw & trim; client-sdk #84 (`95b8d81f`) enforces the canonical minimum at construction; validator-ts #36 rejects confirmed minimum-debt violations (`255b9513`). Verified at core-ts `073c4156`: every action evaluator carries the check.

HISTORY

DUCAT-1978 is protocol-sdk `4e34c23` (16 October), validator-rs `7e23a68`, & guardian-rs `b2843bb`, all on the same day. It arrived inside the October consolidation, with no separate September approval.

WHAT REMAINS

guardian-ts only. Its verify path (`services/guardian-ts/src/lib/verify.ts`) has no `is_valid_unit_balance` call at infra #92 `fafe8220` or the current beta tip — the TS guardian still signs dust-debt transitions that validator-ts then rejects at confirmation. Wire the core predicate into guardian-ts (mirroring the SDK boundary check), or record the TS/Rust signing-gate divergence as accepted. The Rust half stays closed.

NOVEMBER CLOSEOUT

The closeout record's tests reject dust-debt transitions before signing on the TS guardian path, &

withdrawals preserve existing debt including legacy dust. The guardian-ts half that was outstanding is now evidenced; finding closed.

[P2-08] The client SDK did not bind guardian responses to the request

CLOSED ELSEWHERE

Location: client-sdk `src/class/socket.ts`, vault guard helpers, oracle contract fetch; core-ts `create_price_commits` · DUCAT-1979 · ducat-infra #70

ORIGINAL FINDING

High-level helpers checked only that a guardian result was hex with a txid shape; `send()` returned the envelope with no schema. Neither path checked that the returned txid was the PSBT that was sent. Oracle fetch did not call `verify_active_price_contract`. `verify_psbt_output_integrity` existed & had no callers. A buggy or malicious guardian could return a different transaction & the SDK would report success.

CONCERN

The user signs one PSBT & broadcasts another. Output binding is the check that makes "the guardian signed my vault" mean the transaction the wallet displayed.

WHERE THE CODE IS NOW

This finding's files are client-sdk & core-ts, not in the three reviewed pull requests. The September review of ducat-infra #70 is the remediation record: guardian vault responses Zod-parse the envelope, require the returned txid to equal the sent PSBT txid, re-decode the transaction, & check that the guardian signed the planned inputs. Oracle contract fetch calls `verify_active_price_contract`.

HISTORY

The 30 September review accepted #70 & said P2-08 can be marked remediated on merge. The one nit was that a single junk Nostr event fails the whole contract fetch, so a noisy relay can deny service to the client; the review did not block on it. Closing P2-08 against guardian-rs #101 would attach the fix to the wrong repository.

WHAT REMAINS

The Nostr fail-closed nit, if still the client behavior, is availability of oracle fetch, not a substituted transaction.

[P2-09] `get_threshold_price` did not share TypeScript's post-conditions

CLOSED

Location: protocol-sdk `price_bucket.rs:72-88` (`exact_threshold`), `generate_bucket_thresholds`; oracle `relay.rs` · DUCAT-1980

ORIGINAL FINDING

Both stacks computed `ceil(base_price * rate_thold / base_rate)` in `f64`. TypeScript asserted the result was strictly between zero & `base_price`; Rust saturated the cast & did not range-check. Non-finite or negative outputs became 0 on Rust, & a zero threshold is not liquidatable through the oracle path. The recommendation was to keep the ceil formula, return a `Result`, apply `(0, base_price)` on both sides, & reject non-finite outputs.

CONCERN

A zero threshold is a vault the oracle path will never liquidate. An above-spot threshold is a leaf the two stacks will not both commit.

WHERE THE CODE IS NOW

`exact_threshold` rejects a zero base price, a zero threshold rate, or a zero bucket, computes the threshold in `BigUint`, & rounds up when the product does not divide evenly. `generate_bucket_thresholds` drops a result that is zero or at least `base_price`. The oracle relay repeats that bound after `evaluate_bucket`: a threshold of zero or at least `base_price` is `unavailable_signed_contract`.

HISTORY

ducat-infra #79 was reviewed on 30 September & accepted. The nit was that `bucket_liquidation_price` still returned `Ok(0)` for zero collateral, zero debt, or a zero step. The October head does not have that path: `evaluate_bucket` returns `invalid_input`. Ticket 1980's name on validator-rs & guardian-rs is oracle fixtures & complete batches; the formula change sits in `price_bucket.rs` from the 29 September bucket integration.

WHAT REMAINS

The parity class the recommendation asked to keep is a regression test, not an open logic bug. core-ts's asserts were already the behavior Rust was missing.

[P2-10] Decimal scale came from a language-specific printer (filed as P1-02; downgraded to P2 on 26 August)

CLOSED

Location: protocol-sdk `price_bucket.rs:29-54` (`parse_bucket_terms`); core-ts `src/lib/calc.ts` (`get_decimal_count`) · DUCAT-1973

ORIGINAL FINDING

Both stacks counted digits after the decimal point in a printed `f64` . Rust `format!("{step}")` prints `1e-7` as `0.0000001` (7 places); JavaScript `Number#toString` prints `1e-7` (0 places). The same bit pattern can also print 16 places in Rust & 17 in JavaScript. The recommendation was to keep `f64` for rates, stop using a printer as a consensus scale, & reject steps outside a small grid such as `0.01` & `0.001` .

CONCERN

Bucket boundaries are the liquidation price. A one-place disagreement in the scale moves every bucket, so a client-built vault fails validation or becomes unliquidatable against the other stack's tree.

WHERE THE CODE IS NOW

`parse_bucket_terms` no longer prints the step. It tries scales 1, 10, & 100 & keeps the smallest one for which `round(step * scale) / scale` equals the step. Anything else, including `1e-7` & `0.001` , returns `unsupported_step` . `decimal_count` is gone from this head. The local core-ts checkout still has the old helper at `calc.ts:197-206` ; that file is not in these pull requests, & the Rust consensus path no longer calls it.

HISTORY

ducat-infra #77 was reviewed on 30 September & accepted. Both sides reconstructed `step * 10^n` & required an exact round-trip, capped bucket precision at hundredths, & shared `decimal_scales.json` . The review noted that rejecting `0.001` is stricter than the audit's example grid, & that this is acceptable if `0.01` is the protocol step. The same rule is what `ba9be0b` left in `price_bucket.rs` , & it is still the parser at `872fa40` .

WHAT REMAINS

The Rust side matches the recommendation, with the stricter grid. A TypeScript caller that still uses `get_decimal_count` for a consensus scale would reintroduce the split. That helper's removal belongs in core-ts, which these heads do not change.

6.4 Informational / Optimization Findings

[P3-01] TypeScript does not check the committed threshold against the vault

ACCEPTED

Location: core-ts `verify_active_price_contract`, `src/lib/vault/verify/policy.ts:26` (`POLICY_FLAG.BUCKET`); guardian-ts `verify.ts`

ORIGINAL FINDING

The active-contract check asserts the contract is untriggered & that `thold_price` is in `(0, base_price)`. It does not check that the threshold is the vault's collateral-ratio bucket. A hand-built request can bind the wrong leaf. The recommendation was to compute the expected bucket at request time, reject a mismatch, & emit `POLICY_FLAG.BUCKET`.

CONCERN

The wrong leaf liquidates at the wrong price. Rust guardians recompute, so a Rust signing path refuses it. A TypeScript guardian that only checks the shape of the contract will sign it. Mixed stacks are the deployment this finding is about.

WHERE THE CODE IS NOW

The verification-layer check is still absent: `verify_active_price_contract` (core-ts `src/lib/verify/price.ts:118`) authenticates signature, bounds & untriggered state but never compares the committed threshold to the vault's collateral-ratio bucket; guardian-ts authenticates the same way. One change since August: client-sdk #84's selection pipeline (`resolve_signed_price_contracts`) recomputes each oracle's bucket from the vault's actual balances & requires commit-hash equality with the recomputed `thold_price`, so SDK-built requests can no longer bind a mismatched leaf. `POLICY_FLAG.BUCKET` no longer exists — core-ts #27 deleted it with the other unused flags (see P3-05), so the emit-the-flag half of the recommendation is gone by deletion.

WHAT REMAINS

Unchanged in substance: the finding stays open until core-ts or guardian-ts rejects a contract whose threshold is not the position's bucket. The client-side binding narrows the exposure to non-SDK callers; guardian-rs recomputing downstream remains the backstop. A reviewer comment on core-ts #27 asks DUCAT to ticket the verify-layer check or record accepted risk.

NOVEMBER CLOSEOUT

DUCAT's position: the client matches contracts to signed quotes by oracle pubkey (client-sdk #84); guardians receive contracts without quotes & enforce the protocol minimums for collateral ratio & liquidation threshold; custom clients must perform the quote check themselves. The guardian-side bucket recomputation is not being added to TypeScript. SetDev accepts this with the caveat that a non-SDK client is trusted to select the right leaf, bounded by the protocol minimums & by guardian-rs

recomputing downstream.

[P3-02] Oracle commit age is not enforced in the TypeScript guardian

CLOSED ELSEWHERE

Location: guardian-ts signing path; core-ts `policy.ts:25` (`POLICY_FLAG.STALE_PRICE`)

ORIGINAL FINDING

The client SDK drops oracle events whose signed `base_stamp` is older than five minutes, but guardian-ts does not apply that check when signing. `STALE_PRICE` is declared & never emitted. guardian-rs already refuses stale `price_stamp` values before FROST. The recommendation was to call `is_oracle_event_fresh` in guardian-ts before signing, & to emit the flag or delete it.

CONCERN

A stale commit signs a liquidation against a price the oracle has since moved off. The unused flag makes the policy module look like it checks freshness when the evaluators do not.

WHERE THE CODE IS NOW

guardian-ts now enforces freshness before signing. `services/guardian-ts/src/lib/freshness.ts` sets `ORACLE_MAX_AGE_SECONDS = 300` (matching the client fetch default, inclusive) & per oracle contract re-verifies `verify_price_contract` before asserting `is_oracle_event_fresh` on that oracle's own `base_stamp`; stale & future-dated stamps both reject. It runs twice per request: inside `verify_guardian_vault_request` (including the bundled-trim path) & again immediately before signing via `verify_guardian_request_freshness` in the shared `handle_vault_action` flow used by all seven vault routes (open, borrow, repay, withdraw, repo, trim, close). Verified at infra #92 `fafe8220`. core-ts #27 deletes `POLICY_FLAG.STALE_PRICE` — the "emit the flag or delete it" recommendation resolved by deletion, the smaller change. Closed in ducat-infra #92 & core-ts #27, outside the three pull requests reviewed here.

WHAT REMAINS

Nothing on the signing path.

[P3-03] The TypeScript guardian does not bind liquidation reserve payouts

CLOSED

Location: guardian-ts profile-transition checks; protocol-sdk repo & trim validators (Rust reference behavior)

ORIGINAL FINDING

Rust repo & trim validators recompute the liquidation quote & require the tax to be paid to `P2TR(reserve_pubkey)`. guardian-ts checks profile transitions & does not inspect change-output scripts. A crafted PSBT can keep internal amounts consistent while paying the reserve elsewhere. The recommendation was to recompute the quote in guardian-ts & require a change output of at least `reserve_sats` to that script.

CONCERN

The reserve output is the protocol's fee. A PSBT that pays it to the liquidator, with amounts that still add up, is a theft of that fee if a guardian signs it. Rust validators reject it; a TypeScript guardian that never looks at the script will not.

WHERE THE CODE IS NOW

Re-verified at infra #92 `fafe8220`: guardian-ts still checks profile transitions without inspecting change-output scripts; no reserve-output binding exists in its verify path. The Rust requirement stands in protocol-sdk #120's repo & trim validators.

WHAT REMAINS

Unchanged: the finding closes when guardian-ts requires the reserve script, or when production signing is exclusively guardian-rs & guardian-ts is no longer a signing path. The second is a deployment statement, not a diff.

NOVEMBER CLOSEOUT

Closeout tests reject diverted reserve payments & underpayments before signing. The guardian-ts reserve-script binding the finding asked for is now exercised; finding closed.

[P3-04] The strict borrow verifier omits the collateral-ratio floor

CLOSED ELSEWHERE

Location: core-ts `verify_vault_borrow_strict`, `eval_vault_borrow_policy`, `policy.ts:22` (`RATIO_FLOOR`)

ORIGINAL FINDING

`verify_vault_borrow_strict` checks that debt increased & that signatures, guards, & lineage are valid; the ratio floor lives in the policy evaluator. Existing callers call the evaluator, but the function name reads like a complete borrow check. A later integrator calling only the strict function would mint under-collateralized UNIT. The recommendation was composites for open, borrow, repay, & withdraw, or a rename that cannot be misread.

CONCERN

A footgun, not a live bypass on the callers that exist. Under-collateralized UNIT is the failure if the next caller copies the strict export & skips policy.

WHERE THE CODE IS NOW

Unchanged. core-ts #27 threads a required transaction through the repo & trim composites (see the DUCAT-2054 settlement work) but adds no composites for open, borrow, repay or withdraw; the strict export shape is the same.

WHAT REMAINS

The composites, or the rename.

NOVEMBER CLOSEOUT

core-ts `d9fce5c`: complete signing checks (strict + policy composites) now exist for open, borrow, repay & withdraw, & the client & TS guardian call them. Closed in core-ts, outside the three reviewed heads.

[P3-05] Borrow & withdrawal limit helpers & three policy flags are unused

CLOSED ELSEWHERE

Location: core-ts `verify_borrow_limits`, `verify_withdrawal_limits`, `policy.ts:25-28`

ORIGINAL FINDING

The two limit helpers have no callers. `STALE_PRICE`, `BUCKET`, & `FEE` are declared & never emitted. The recommendation was to wire the flags or delete the dead surface.

CONCERN

A reviewer reads the flag list & believes freshness, bucket match, & fee policy are enforced. They are not. That is how P3-01 & P3-02 stay invisible.

WHERE THE CODE IS NOW

Deletion shipped — the smaller change the finding preferred. At core-ts `073c4156`, `POLICY_FLAG` is exactly the six emitted codes (`DIRECTION`, `RATIO_FLOOR`, `RATIO_UNAVAILABLE`, `REPAY_WORSENED`, `VALUE_FLOOR`, `LIQ_FORMULA`); `STALE_PRICE`, `BUCKET` & `FEE` are gone, & `verify_borrow_limits` / `verify_withdrawal_limits` are deleted with no callers. The reviewer's concern inverts cleanly: with the flag list now equal to the emitted set, the module no longer advertises checks it does not run. The checks those flags named are tracked on their own merits: freshness now enforced — P3-02; bucket match accepted as a client-side check — P3-01. Closed in core-ts #27, outside the three pull requests reviewed here.

WHAT REMAINS

Nothing.

[P3-06] Vault script parsers accept non-curve public keys CLOSED ELSEWHERE

Location: core-ts `parse_cosigner_script`, `parse_liquidation_script`

ORIGINAL FINDING

Both parsers reject only the all-zero 32-byte string. Off-curve keys fail later, at signature verification & at Bitcoin consensus. `assert_bip340_pubkey` already exists. The recommendation was to call it in both parsers.

CONCERN

A transaction that parses & then fails on chain wastes a signing round & looks like a guardian bug. It does not move funds to an off-curve key.

WHERE THE CODE IS NOW

Both parsers call `assert_bip340_pubkey` (core-ts `a2303c80`, 16 October): an off-curve key now throws a typed `ProtocolInvalidError` ("invalid vault script public key") at parse time instead of failing at signature verification or on chain. Recovery paths retain the typed rejection (`2b69736f`). Verified in the #27 diff at `073c4156`. Closed in core-ts #27, outside the three pull requests reviewed here.

WHAT REMAINS

Nothing.

[P3-07] Verifier exports do not show which checks are complete

CLOSED ELSEWHERE

Location: core-ts verify module exports for open, borrow, repay, withdraw

ORIGINAL FINDING

Trim & repo export composites that run strict checks & then policy. Open, borrow, repay, & withdraw export only the strict half. A caller that copies the borrow export does not get policy. The recommendation matches P3-04.

CONCERN

Same footgun as P3-04, spread across four actions instead of borrow alone.

WHERE THE CODE IS NOW

Unchanged in core-ts: open, borrow, repay & withdraw still export the strict half without composites (verified at `073c4156`). Rust validation on #120 remains per-action `validated.rs` modules, a different export shape, & was not the finding.

WHAT REMAINS

The composite exports, or the rename.

NOVEMBER CLOSEOUT

Same commit as P3-04 (core-ts `d9fce5c`): all four actions export composites, so the export surface no longer hides which checks are complete. Closed in core-ts.

[P3-08] `hasher_to_scalar` could panic & disagreed with upstream

CLOSED

Location: guardian-rs `vendor/frost-secp256k1-tr/src/lib.rs:187-191`

ORIGINAL FINDING

The pinned fork unwrapped `ScalarPrimitive::new` on a SHA-256 digest. Landing at or above the curve order is about a 2^{-128} event & not grindable, but the unwrap sat on every sign, verify, & tweak. Upstream uses `Scalar::reduce`; for any input at or above the order the two would disagree, a second silent incompatibility of the same family as P2-01.

CONCERN

A panic on a path that has never fired, plus a future bump that changes challenge & tweak scalars for an input nobody has produced.

WHERE THE CODE IS NOW

Line 187 is `Scalar::reduce(U256::from_be_slice(&digest))`, & `hasher_to_scalar` calls it. The unwrap is gone. For every digest below the order, reduce & the old constructor agree, so existing signatures do not move.

HISTORY

Landed with the FROST 3 vendor move (`d43ab38`, 18 October), the same consolidation as P2-01.

WHAT REMAINS

Nothing on the panic. The pin is a vendored tree, which is the right place for a consensus-sensitive fork, & P2-01's enum is what keeps a later vendor update from flipping tweak meaning.

[P3-09] Participant final-signature verification was unreachable

CLOSED

Location: guardian-rs `session/handler/participant.rs:43-46, 133-160`, `session/worker.rs:106`

ORIGINAL FINDING

The worker treated `ShareSent` as terminal, so participants left after sending their share. Final-signature handling ran only in `CollectingSignatures`, a state participants never enter. The documented defense-in-depth check did not run. The recommendation was to keep the participant alive until `FinalSignature` or timeout, or to delete the unreachable handler.

CONCERN

A coordinator that broadcasts a signature over a different sighash would not be caught by the participant. The chain would still reject the spend; the missing check is the one that fails the session before anyone broadcasts.

WHERE THE CODE IS NOW

On `9f0e566`, `participant.rs:43-46` matches `(ShareSent, FinalSignature)` & calls `participant_handle_final_signature`, which verifies the signature against the local group key, the local signing mode, & the sighash stored from the validated payload, then sets `Completed` & clears secrets. `worker.rs:106` treats only `Completed` & `Failed` as terminal. The short status pass called this open; that was wrong on this head.

HISTORY

The local working tree still has the August shape because it is not `9f0e566`. The change sits in the signing-session work that also carries DUCAT-1990, 1991, & 1993. The new handler's comment cites DUCAT-1864 & extends sighash pinning to the aggregate.

WHAT REMAINS

A participant that never receives `FinalSignature` depends on the session timeout already used for the rest of the state machine. That was not a new gap.

[P3-10] The coordinator accepted shares from outside the signing package

CLOSED

Location: guardian-rs `coordinator/handler.rs:335-337` · DUCAT-1990

ORIGINAL FINDING

Any share whose sender had passed peer-identifier binding was counted, including guardians not selected for the package. Aggregation then failed & the session aborted. The recommendation was to reject shares whose sender is not in `signing_package.signing_commitments()`.

CONCERN

A guardian on the allowlist but not in this package can stall a signing round. They cannot produce a signature over a different message, because the sighash is pinned locally.

WHERE THE CODE IS NOW

`7b86927` (24 September), "enforce selected signing participants." A `SignatureShare` is dropped unless `guard.is_selected_signer(&from)` & that identifier does not already have a share.

HISTORY

DUCAT-1990 landed in September & was carried forward. The 26 October re-read did not reopen this bug; it found a different one on the line that runs after the selected-signer check (Section 7, `handler.rs:348`).

WHAT REMAINS

The finding as written is closed. The residual is that `from` is the identifier inside the message & is stored against the authenticated peer with `or_insert`. That can abort a session; it cannot aggregate a valid signature.

[P3-11] The coordinator panicked on its own share if the registry was disabled

CLOSED

Location: guardian-rs coordinator local-share signing · DUCAT-1991

ORIGINAL FINDING

Signing the coordinator's own share used `.expect()`. With only `--allowed-peers`, an allowlisted peer can overwrite the coordinator's commitment, local `sign` fails, & the session worker panics. The recommendation was a session error instead of a panic, a refusal to overwrite the local identifier's commitment, & a fail-closed startup when allowed-peers is set without a registry.

CONCERN

A panic in the session task drops that signing round &, if uncaught higher up, the process. The third bullet is a separate control, tracked as P3-18.

WHERE THE CODE IS NOW

`67a30c3` (24 September), "prevent coordinator signing panics," with tests in `0f43c4b` & `1eb8794`. The selected-signer check from DUCAT-1990 refuses an out-of-package share before the coordinator signs its own.

HISTORY

1990 & 1991 landed the same day as a pair. October's later commits do not revert them.

WHAT REMAINS

Startup with `--allowed-peers` & no registry is P3-18. This finding's panic is closed.

[P3-12] The libp2p identity is derived from the FROST signing share

ACCEPTED

Location: guardian-rs `guardian-exe/src/keygen.rs:112`, `config.rs:548, 603`

ORIGINAL FINDING

`create_p2p_keypair_from_frost` sets the Ed25519 peer key to `SHA256(tag ‖ identifier ‖ signing_share)`. Share rotation changes `PeerId`. The same key is used for gossipsub & Noise. Zeroizing the share does not cover the derived key. The recommendation was an independent secret, or an HKDF with an explicit domain, & a comment if determinism is intentional.

CONCERN

Operational coupling: rotating a FROST share silently rotates the peer identity, & the long-lived Noise key is a function of the signing share.

WHERE THE CODE IS NOW

The derivation & both call sites are unchanged. DUCAT-1993 erases session secrets but does not change keygen; the FROST 3 move does not introduce a separate peer secret.

WHAT REMAINS

The derivation, or a comment that the coupling is intentional & that share rotation is a peer-identity rotation.

NOVEMBER CLOSEOUT

DUCAT is keeping the share-derived peer identity. guardian-rs `80b2e47` documents the derivation (SHA-256 with the dedicated domain string `libp2p-keypair-derivation-v1:`, participant identifier & signing share), that share changes require coordinated PeerId / registry updates, & that erasing a share does not erase the derived network key. That is the "comment that the coupling is intentional" alternative from the recommendation; SetDev accepts it.

[P3-13] FROST nonces were not zeroized on failure paths

CLOSED

Location: guardian-rs `participant.rs`, `worker.rs` (`send_completion_result`) · DUCAT-1993

ORIGINAL FINDING

Nonces were zeroized after a successful `sign()`; on error or timeout they lived until the session was dropped. No nonce-reuse path was found. The recommendation was to zeroize on teardown & adopt `ZeroizeOnDrop` with the FROST upgrade.

CONCERN

A nonce that survives a failed round is the input to a nonce-reuse bug if a later path signs again with it. Hygiene on the hottest secret in the process.

WHERE THE CODE IS NOW

`participant_handle_signing_package` takes nonces with `secrets.take_nonces()` & calls `g.secrets.clear()` after the share is produced. `participant_handle_final_signature` clears again on completion. `send_completion_result` clears secrets before publishing, including participant sessions with no completion sender. `3ffb9b3` (6 October), "revoke & erase owned session secrets," with `45e908b`.

HISTORY

The FROST 3 commit (`d43ab38`) is titled with one-use nonces as well as checked tweaks. The two changes are the recommendation's two bullets.

WHAT REMAINS

A process crash still leaves whatever the OS keeps of the address space. That is outside the finding.

[P3-14] Huffman taproot builders were public

CLOSED

Location: protocol-sdk `connector.rs`, `taproot.rs:51`, `script/mod.rs:65` · DUCAT-1994

ORIGINAL FINDING

Helpers used `TaprootBuilder::with_huffman_tree`. Huffman & sorted-balanced trees match for one or two leaves & diverge for three or more. Production callers passed one script; a future caller passing several would build a different root from the TypeScript client. The recommendation was to route through `merkleize_sorted_balanced`, or constrain the API to one script.

CONCERN

A merkle-root split means the vault address the client derived is not the address the Rust builder funded.

WHERE THE CODE IS NOW

`7397474` (24 September), "restrict connector trees to one script." The helpers now build a taproot address from a single script leaf; `tests/vault_output_parity.rs:71` records the Huffman history & `tests/connector_taproot.rs` documents the single-leaf case.

HISTORY

The recommendation's second option. validator-rs & guardian-rs took that SDK on 24 September (`6e2260e`, `f5300d9`), together with DUCAT-1995.

WHAT REMAINS

A new caller that needs three leaves has to go through the sorted-balanced path on purpose. The public Huffman entry is no longer the connector API.

[P3-15] The breached-commit filter existed only on the Rust tree builder

CLOSED ELSEWHERE

Location: protocol-sdk `liquidation_hashes`; client-sdk `script.ts` · DUCAT-1995

ORIGINAL FINDING

Rust dropped commits whose `thold_price` was not strictly below `base_price` before building leaves; TypeScript included every commit. The two agreed only because upstream validation already rejected breached commits. The recommendation was to mirror the filter in TypeScript, or assert the invariant at tree-build time on both sides.

CONCERN

Leaf sets that differ by one commit produce different merkle roots — the same failure mode as P3-14.

WHERE THE CODE IS NOW

`05bd199` (5 October), "make defensive tree filtering consistent," is the Rust side agreeing with itself. The TypeScript mirror has since landed: client-sdk `ac672e4c` (15 October) aligns pledged-vault tree construction & cooperative/liquidation spend filtering with the Rust SDK — same positive, below-base commitments on both sides. The shared fixture `test/fixtures/vault/tree_parity.json` is checked by the client's `tree-parity.test.ts` **and** the Rust SDK's `breached_tree_parity` test, so the filter is pinned bidirectionally, not just mirrored. TS half closed in client-sdk #84, outside the three pull requests reviewed here.

WHAT REMAINS

Nothing.

[P3-16] `TAPLEAF_VERSION` is `0x51` & unused

CLOSED ELSEWHERE

Location: client-sdk `src/const.ts`

ORIGINAL FINDING

Nothing reads the constant; tapscript leaves use `0xc0`. Wiring it into leaf encoding would change every leaf hash relative to Rust. The recommendation was to delete or correct it.

WHERE THE CODE IS NOW

The constant is corrected, not deleted: client-sdk `src/const.ts:51` now exports `TAPLEAF_VERSION = 0xc0` (`a0cbdb4b`), with a commitment regression & negative control pinning the value. The caveat from the original finding holds & was checked: no second copy still says `0x51` — the remaining `0x51` in `OUTPUT_CODES` (`const.ts:31`) is an OP_RETURN output code, unrelated to leaf encoding. Closed in client-sdk #84, outside the three pull requests reviewed here.

WHAT REMAINS

Nothing.

[P3-17] Merkle parity is pinned in one direction, & the fixture generator is missing

CLOSED ELSEWHERE

Location: protocol-sdk `tests/vault_output_parity.rs`; `dev/gen-vault-parity-fixture.mjs` (absent); validator-ts `@vbyte/btc-dev` range

ORIGINAL FINDING

The Rust test pins TypeScript merkle fixtures; the TypeScript tree has no matching pin, & the generator is absent. validator-ts allows `^2.1.0` where client-sdk pins 2.1.0. A TypeScript change can move the tree without a test, & a caret range can move the validator's merkle library.

CONCERN

P3-14 & P3-15 are the ways the trees diverge. This is how that divergence would ship without a red test on the TypeScript side.

WHERE THE CODE IS NOW

All three asks landed. The generator is restored & moved into the test suite (`4f2c94d3`, `57bf6fcf`); `test/fixtures/vault/tree_parity.json` + `tree-parity`, `output_parity` & `liq-thold-parity` tests pin the TypeScript tree, & the same JSON is checked by the Rust SDK's parity test — the pin is now bidirectional. validator-ts #36's `package.json` pins `@vbyte/btc-dev` to exactly `2.1.0` (caret range removed, verified at `9e26c27e`). Closed in client-sdk #84 & validator-ts #36, outside the three pull requests reviewed here.

WHAT REMAINS

Nothing.

[P3-18] libp2p identity binding is optional without a peer registry

CLOSED

Location: guardian-rs `guardian-exe/src/config.rs:591, 621-623`, `args.rs:247-252`

ORIGINAL FINDING

When only `--allowed-peers` is set, identifier binding is skipped & any allowlisted peer may claim any FROST identifier. Session integrity still holds through sighash pinning. The shipped infra always sets a registry. The recommendation was to fail closed at startup in that configuration.

CONCERN

An allowlisted peer that can claim another signer's identifier is the setup for the residual at `handler.rs:348`. Fail-closed startup removes the configuration in which that is possible.

WHERE THE CODE IS NOW

`config.rs:591` loads a registry when a path is set & falls through when it is not. Lines 621–623 warn if both flags are set & then use the registry; the warn is not a refusal of allowed-peers alone. No commit changes `load_peer_registry_and_allowed_peers` into a hard error.

WHAT REMAINS

Return an error at startup when allowed-peers is set & the registry path is empty. The shipped compose can keep working.

NOVEMBER CLOSEOUT

Trusted peer bindings are now mandatory: a participant cannot claim another signer's identifier, including through relays (guardian-rs #101 reply). The allowed-peers-without-registry configuration the finding described no longer exists; finding closed. See also R-06.

[P3-19] Validator statistics wrapped on raw addition

CLOSED

Location: validator-rs `storage/rocksdb/batch.rs:1866-1867, 1889-1890` · DUCAT-1998

ORIGINAL FINDING

Wrapping `+=` for global volume & lock deltas corrupts reported totals; it does not change consensus. The recommendation that shipped was saturating arithmetic on the stats accumulators.

CONCERN

A wrapped total looks like a supply or volume figure that went backward. Vault acceptance does not consult these counters.

WHERE THE CODE IS NOW

Both sites use `saturating_add`. `9918289` (6 October), "bound validator statistics on the 2054 storage foundation," & `a8c051a` reconcile with recovery accounting. The head `379ff54`, "verify live statistics bounds & record mint admission gap," is the regression test.

HISTORY

The August draft had a false sentence about vault volume already saturating; it was corrected before the report was treated as final. The implementation matches the corrected recommendation.

WHAT REMAINS

Saturation clamps a total that would exceed the integer width; it does not page an operator. Consensus paths use the checked addition from P2-06, not this saturating add.

[P3-20] The validator HTTP API is unauthenticated

CLOSED

Location: validator-rs public HTTP API · DUCAT-1999

ORIGINAL FINDING

The API is read & compute, with pagination caps & no in-process per-client throttle. A public deployment would depend on something in front of the process. The recommendation was to document that assumption.

CONCERN

An open read API is a CPU & disk hose. It is not a key store, which is why authentication was not asked for here.

WHERE THE CODE IS NOW

`4188fd2` (24 September), "limit public HTTP requests." `759de83` returns 429 & checks the rendered proxy policy; `df1ace4` covers the throttle in tests. guardian-rs `a705ffa` selects the same protection. This is a stronger change than the recommendation.

WHAT REMAINS

Anyone who can reach the port can still call it, up to the limit. That matches the original scope.

7. Residuals from the 20–26 October Re-read

These are not August IDs. They are what remained after the comment pass on the three heads, once comments that did not survive a second reading were removed.

Removed from the list — presign stack check. `crates/bitcoin-consensus/src/presigning.rs` on protocol-sdk #120 (`017c48f` , DUCAT-2059) accepts a liquidation witness when the guardian signature slot is missing & the rest of the stack is the preimage, the script, & the control block. Builders stage `[preimage, script, control]` , which fails the check. guardian-rs `request.rs:89-92` (`8aa62e0`) inserts 64 zero bytes at the front when a taproot witness has three elements, & only then calls authorize. The witness that reaches the checker is the shape the checker requires, & a bad witness is rejected. The first draft treated the disagreement as a bug; the re-read took it off the list.

R-01 Mint position · validator-rs `mint_identity_tests.rs:86`

`379ff54` adds `mint_identity_requires_candidate_position` & ignores it with "known mint-position admission gap; see `docs/statistics-bounds.md` ." The test calls `validate` & expects an error when the same mint commitment is checked at another height or index. `validate` still returns success. The file checks typed validation, not mined-chain acceptance; it does not run block indexing, witness checks, or `SetUnitReserveUtxo` . An earlier draft said indexing then installs the skipped mint as the reserve — that write exists on `dev` for an already-valid `UnitMint` , & this ignored test does not reach it. The accurate comment: validation still accepts the commitment at the wrong position, & the ignore is the line the pull request adds.

NOVEMBER CLOSEOUT

Fixed. protocol-sdk `69f0e54` & validator-rs `dedd1d4` check mint position & identity against the anchor & store the result for reuse; unrelated or duplicate mints can no longer reset the reserve or credit UNIT. The ignored test is no longer the record of a gap. DUCAT tracks this as SEC-06. Broader UNIT accounting remains a separate workstream.

R-02 Authorize reports a non-action as valid · `tx_validate.rs:312`

For a transaction that is not a recognized Ducat action, `valid` is `guardian.is_some()`. That flag means the witness check passed, not that the transaction was accepted as a protocol action. This handler does not insert the three-element placeholder before the SDK check, so a raw builder PSBT & a guardian-adjusted PSBT do not take the same path through `/api/tx/authorize`. Callers that treat `valid: true` as "this is a Ducat transaction" are reading a witness result.

NOVEMBER CLOSEOUT

Fixed. `/api/tx/authorize` results now distinguish `ducat_authorized`, `bitcoin_authorized` & `rejected`, so a witness-only pass is no longer reported as a protocol acceptance.

R-03 Latest vault hides a burn · `vault_latest.rs:105`

The `Broken` arm returns `VaultNotFound`, so a stored burn 404s. A voluntary close still resolves. The burn row keeps `action: None` & the burn txid; the handler does not read that txid. `8b0434c` documented the sibling lifecycle route as the place the break is returned. Latest itself still answers not-found.

NOVEMBER CLOSEOUT

Fixed. The latest-vault endpoint returns the last valid profile with `state: broken` when the vault's UTXO chain is broken, instead of 404.

R-04 History skips the burn row · `vault_history.rs:315`

```
if record.broken.is_some() {
    continue;
}
```

The burn is one of the stored rows; it has no action, & `txid` is the transaction that spent the vault. `continue` skips it before that txid is rendered. `/api/vault/{id}/lifecycle` reads the same row & returns `spend_txid`. A client that only calls history cannot see how the vault was burned. This does not drop the burn from storage & does not accept a bad transaction.

NOVEMBER CLOSEOUT

Resolved as designed. History lists valid actions; the break & any recovery are exposed on the lifecycle endpoint. The burn row stays out of History by design, & the spend txid is reachable through lifecycle. SetDev accepts the split.

R-05 Three liquidation errors do not abort the block · `state.rs:1593`

`liquidation_context_error` aborts on `InvalidLiquidationThreshold` & on a list of structural repo & trim errors. It does not match `LowLiquidationThreshold`, `LiquidationRatioUnavailable`, or `InvalidCommitCount`. Those three were named in the October review & are still absent. The repo or trim is stored invalid & not applied; the spent vault is marked broken; the block continues. `8b0434c` added tests that those errors do not abort, which records the behavior rather than changing it. Whether that is the intended split between "abort the block" & "mark the vault broken" is the open question.

NOVEMBER CLOSEOUT

Resolved as designed. For confirmed liquidations, a threshold below the minimum produces a validator warning & the vaults in the transaction remain active; guardians still reject the policy violation at signing. Malformed transactions remain invalid, with the vaults recoverable. Missing context or storage failures abort the block. That answers the open question on the abort / mark-broken split; SetDev accepts it.

R-06 Peer identifier is not bound to the authenticated peer · `handler.rs:348`

After a `SignatureShare` passes `is_selected_signer(&from)`, the sender is stored with `participant_identities.entry(source_peer_id).or_insert(from)`; the same bind happens for a fresh commitment around line 245. `from` is the identifier inside the message & is not compared to an identifier this peer already registered. `SessionFailed` then treats the stored identifier as the sender. A peer can submit one share, or one fresh commitment, under another selected signer's identifier & abort that session. The share cannot be aggregated into a valid signature. This is the availability issue P3-10's fix did not cover; P3-18's startup refusal would remove the configuration where an allowlisted peer can attempt it.

NOVEMBER CLOSEOUT

Fixed. Mandatory trusted peer bindings (see P3-18) mean a share or commitment is only accepted under the identifier bound to the authenticated peer; the identifier-hijack abort is no longer reachable.

R-07 First price commit · `repo.rs`, `trim.rs`

`repo.rs` & `trim.rs` still take the breached commit from `price_commits.first()`. `bf66f11` & `9bc24ac` (14 October) fixed a different bug on those commands: they now take `vault_min_amount` & `liquidation_thold` from the current protocol. The first-commit selection was not changed, so the review UI will not accept a comment on it. The wallet opens the first commit rather than the commit this oracle can breach. That predates guardian-rs #101.

NOVEMBER CLOSEOUT

Fixed. guardian-rs `6e76f02`: repo & trim select the supplying oracle's authenticated contract from the target vault & the matching breach proof, rather than `price_commits.first()`. Missing or ambiguous matches reject.

8. Other Tickets on These Heads

These commits are part of the three pull requests & are not one of the August finding IDs. They are included so the remediation record matches the commit list, not only the August index.

TICKET	DATE	COMMIT	WHAT IT IS
DUCAT-1975	October	`0d9eadd`, `ae88388`	Authenticated Rust contract selection, & cleared transfers that do not require an oracle contract.
DUCAT-1980	October	guardian `3985bab`, validator `f398a9d`	Complete oracle batches before publication, & the fixtures that go with them.
DUCAT-1983	October	`456c009` + recovery series	Permissionless recovery state, accepted debt, & profile identity across all three repositories.
DUCAT-1984	October	`70c7f15`	Oracle contracts bound to vault outputs in tests.
DUCAT-1988	October	guardian `6498b4e`	FROST digest boundaries, signing modes, & nonce probes.
DUCAT-1994	24 Sep	`7397474`	One-script connector trees. See P3-14.
DUCAT-1995	5 Oct	`05bd199`	Defensive tree filtering on the Rust side. See P3-15.
DUCAT-2005	Sep–Oct	`6638781`, `83e9ae8`	Liquidation settlement uses the signed base price. The threshold authorizes liquidation; it is not the settlement price.
DUCAT-2030	September	`951ace7`	Ratios & liquidation use net collateral.
DUCAT-2049	September	`5b35c22`	Canonical ratio consumers on the signing path; a guardian-local ratio gate was deleted so signing uses `validate_transactions`.
DUCAT-2052	October	`16c1d0c`	Reserve counted in native account bootstrap.
DUCAT-2053	29 Sep	`ba9be0b`	Independent oracle buckets. The commit that introduced `price_bucket.rs`.

TICKET	DATE	COMMIT	WHAT IT IS
DUCAT-2054	to 20 Oct	heads `872fa40`, `379ff54`, `9f0e566`	Storage, bootstrap, diagnostics, & the recovery merges the other tickets sit on.
DUCAT-2059	19 Oct	`017c48f`, `8aa62e0`, `9c3d511`	Presigning authorization separated from completed validation, & the witness placeholder described in Section 7.

9. Conclusion

The October consolidation resolved the numeric, signing-mode, & session-hygiene findings from the August audit on the Rust side, & the service-side hardening (request limits, statistics bounds, deployment policy) matched or exceeded what was recommended. The November closeout finished the TypeScript half & the seven residuals. Every August finding is now closed or accepted; no finding remains open or partial.

26

CLOSED

Against these heads (15): P2-01, P2-02, P2-03, P2-06, P2-09, P3-08, P3-09, P3-10, P3-11, P3-13, P3-14, P3-18, P3-19, P3-20, P2-10 (was P1-02). Outside these PRs (11): P2-07, P2-08, P3-02, P3-03, P3-04, P3-05, P3-06, P3-07, P3-15, P3-16, P3-17.

3

ACCEPTED DESIGN CHOICES

P2-05 (embedded validator required on mainnet), P3-01 (quote bucket check stays client-side), P3-12 (share-derived peer identity, documented).

0

LEAVE OPEN

None.

Deployment posture: P2-05 is accepted on the basis that mainnet requires the embedded, in-process validator. Any future configuration that re-enables a remote validator outside regtest reopens the finding.

The seven residuals in Section 7 are resolved: R-01, R-02, R-03, R-06 & R-07 by code; R-04 & R-05 as confirmed design. Broader UNIT accounting remains a separate workstream outside this engagement. The presign stack check is not an open defect.

This report reflects the state of the referenced commits at the time of review (October heads plus the November 2026 closeout commits). It is not a guarantee of the absence of defects & should be read alongside the August 2026 audit & the project's own ticket history.