

PoRepMarket v2 Security Review

Response to the security audit conducted by Composable Security.

1. Executive summary

How to read this revision. This document was revised after the post-audit remediation round and describes main at v2.0.0-rc. The current state has been re-verified against that commit for every finding, including the findings whose status did not change, so it always reports the implementation as it stands today rather than at the time of the original response. Post-audit remediation records what changed since that response and, where the delivered fix differs from the approach first proposed, why. Planned action records what is still outstanding.

Deployment context. PoRepMarket v2 will be deployed as a new system using new proxy instances. It is not intended to reuse v1 proxy storage. This context is central to the team's position on H01, while strict upgrade-safe storage discipline remains mandatory for every subsequent v2 upgrade.

Remediation since code freeze. The repository contains material fixes for stalled deal cleanup, evidence activeness, partial refresh snapshots and unprovided evidence recovery. A subsequent post-audit remediation round has since been merged and tagged as v2.0.0-rc. It closes the posted-size upper bound, cross-deal claim accounting, client-scoped manifest assignment, the latency attestation gate, the bounded and explicitly initialized activation padding, the early-termination cap, the named settlement deferral errors, a centralized access manager with a delayed two-step admin transfer, paginated view functions, paused-provider offer checks and the remaining informational cleanups. The full contract test suite, 492 tests across 15 suites, passes on that commit.

DataCap transition. The Filecoin network is planning a FIP to remove DataCap. In the current product context, the project does not treat DataCap as an economically high-value token. DataCap-specific findings are therefore assessed in light of the expected protocol transition, while issues affecting payments, provider capacity or deal integrity remain in scope for remediation.

Planned work. Items marked Post-audit resolved were open when the audit engagement closed and have since been closed; each such finding records the change and any deviation from the approach originally planned. No finding remains in the Planned, Under review or Decision pending state. Two findings, L01 and L05, are marked Planned for post-v2: both are acknowledged, both need an economic mechanism rather than a contained contract change, and both are scheduled for a release after PoRep Market v2. L06 is accepted in its current form, for the reasons recorded with the finding.

Status legend

Status	Meaning
Resolved	The relevant defect has been removed or the intended behavior is implemented on main.

Status	Meaning
Post-audit resolved	The finding was still open when the audit engagement closed and has since been closed, either by a change verified on main, by a remediation outside the contracts, or by a documented decision to retain the current behaviour. The detail, and any deviation from the approach originally planned, is described in the finding.
Partially resolved	A material part of the risk is mitigated, but a remaining condition or merge is required.
Planned	An internal task has been created for near-term remediation.
Planned for post-v2	The finding is acknowledged, but a durable remediation requires a larger feature rather than a contained contract change, and is scheduled for a release after PoRep Market v2.
Accepted design / risk	The behavior is intentional or accepted in the stated deployment and product context.
Under review	The remediation is implemented in an open pull request but has not yet been merged and verified on main.
Decision pending	The finding is acknowledged, but a product or operational decision is required before implementation.
Not applicable	The affected code path has been removed from the current implementation.
No change planned	The project accepts the current implementation for the stated reasons.

Status overview

Finding	Severity	Status
H01	High	Accepted design
H02	High	Accepted transitional risk
M01	Medium	Planned → Post-audit resolved
M02	Medium	Under review → Post-audit resolved
M03	Medium	Partially resolved → Post-audit resolved
M04	Medium	Resolved
M05	Medium	Accepted design
M06	Medium	Resolved
M07	Medium	Resolved
M08	Medium	Resolved
M09	Medium	Planned → Post-audit resolved
L01	Low	Decision pending → Planned for post-v2
L02	Low	Planned → Post-audit resolved
L03	Low	Planned → Post-audit resolved
L04	Low	Under review → Post-audit resolved
L05	Low	Planned → Planned for post-v2
L06	Low	Partially resolved → Accepted design
L07	Low	Planned → Post-audit resolved
L08	Low	Partially resolved → Post-audit resolved

Status overview (continued)

Finding	Severity	Status
R01	Informational	Planned → Post-audit resolved
R02	Informational	Not applicable
R03	Informational	Planned → Post-audit resolved
R04	Informational	Not applicable
R05	Informational	Planned → Post-audit resolved
R06	Informational	Planned → Post-audit resolved
R07	Informational	Planned → Post-audit resolved
R08	Informational	No change planned
R09	Informational	Planned → Post-audit resolved
R10	Informational	Planned → Post-audit resolved
R11	Informational	Planned → Post-audit resolved
R12	Informational	Partially resolved → Post-audit resolved

2. High-severity findings

[FIDL-fcd6bac3-H01] Storage namespace retained across incompatible struct restructure

Status: Accepted design | Severity: High

Our perspective. PoRepMarket v2 is intended to be deployed as a new system with new proxy instances rather than as an in-place upgrade of the v1 deployment. Under that deployment model, the incompatible v1-to-v2 storage layout does not create a state-corruption path because no v1 state is reused by the v2 proxies.

Current state. The project accepts the auditor's underlying storage-layout principle. Any future upgrade of the v2 contracts must preserve the established ERC-7201 namespace, retain existing field order and types, and append new fields only at the end of the relevant storage structure. The v2 implementation must not be used as a direct upgrade target for an existing v1 proxy.

[FIDL-fcd6bac3-H02] Client can drain the adapter DataCap allowance

Status: Accepted transitional risk | Severity: High

Our perspective. The forthcoming Filecoin FIP removes DataCap, so this allowance issue will not exist in the resulting protocol design. In the current deployment context, DataCap is not treated as an economically high-value token.

Current state. The team accepts the limited transitional exposure and does not plan to redesign the allowance model before DataCap is removed. The upper-bound remediation is now merged on main: submitDataCapBatch rejects a batch whose cumulative evidence would exceed requestedSizeBytes increased by the configured activation padding, so oversized evidence can no longer be accepted for deal accounting, which further limits the practical impact. After the protocol transition, the affected allowance flow will no longer exist.

3. Medium-severity findings

[FIDL-fcd6bac3-M01] An unset latency attestation is scored as meeting the SLA

Status: Post-audit resolved | Severity: Medium

Our perspective. A zero latency value currently has two possible meanings: an actual zero measurement or an omitted measurement. Treating both as the best possible latency is ambiguous and can overstate the score.

Current state. The remediation is implemented on main. SLIOracle.setSLI rejects an attestation whose latencyMs is 0 or equal to the LATENCY_UNMEASURED sentinel, type(uint16).max, with InvalidLatencyMs, so an unset or invalid measurement can no longer be stored and therefore cannot be scored as satisfying the SLA.

Post-audit remediation. The rule is enforced a second time at the scoring layer as defense in depth: SLIScorer.calculateScore counts a required latency dimension as met only when the attested value is neither 0 nor the sentinel, so an attestation written before this change is still never scored as satisfying the SLA. SharedTypes documents the sentinel and the mandatory-measurement rule on the SLIThresholds struct.

[FIDL-fcd6bac3-M02] Posted data size has no upper bound

Status: Post-audit resolved | Severity: Medium

Our perspective. The accepted deal size must remain the upper bound for evidence accounting and billing. Allowing a client to post more than requested can inflate the payment rate and interfere with capacity reserved by other deals.

Current state. The remediation is merged on main. submitDataCapBatch derives maxAllocatedBytes from the deal's requestedSizeBytes and the configured activation padding and reverts with AllocatedBytesExceededRequestedSize when the deal's already recorded allocatedBytes plus the new evidence in the batch would exceed that ceiling. The check covers new allocations and claim extensions alike, because both contribute to the same newEvidenceBytes total. The lower activation threshold is aligned with the same padding: finishDataCapPosting calls _ensureAllocationCoverage, which requires allocatedBytes to reach requestedSizeBytes reduced by the activation padding.

Post-audit remediation. One deviation from the description given when the finding was first answered: the ceiling is requestedSizeBytes increased by the activation padding rather than requestedSizeBytes exactly, because allocation sizes are padded piece sizes and can legitimately exceed the requested figure by that tolerance. With MAX_DEAL_ACTIVATION_PADDING now bounded at 2000 basis points under R01, the overshoot is bounded at 20 percent, and the value assigned at initialization is 10 percent.

[FIDL-fcd6bac3-M03] Evidence refresh state is stale and permissionlessly resettable

Status: Post-audit resolved | Severity: Medium

Our perspective. Permissionless maintenance is desirable for liveness, but a caller must not be able to erase valid progress or combine observations across an unbounded refresh period.

Current state. The remaining refresh changes are merged on main. `refreshEvidenceStatus` records `partialEvidenceRefreshEpoch` only when a new sweep begins, that is when `checkedClaims` is zero, so a caller can no longer push the staleness window forward by repeatedly touching an in-progress sweep. Progress is reset only when the previous sweep completed, when an in-progress sweep is older than one month, or when the deal's termination revision changed mid-sweep, and the reset now clears `pendingActiveClaimedBytes` rather than the last completed `activeClaimedBytes` snapshot. Claim bookkeeping is scoped by deal: `_dealByAllocationOrClaimId` records the owning deal for every allocation and claim id, `_claimDealIds` routes an early termination to its deal, and `_terminationRevisions` is incremented per deal so a termination invalidates only the affected deal's sweep. Claims are always read from `VerifReg` under the deal's own provider.

Post-audit remediation. The staleness comparison was also corrected. The audited code compared `partialEvidenceRefreshEpoch` against the current epoch plus one month, a condition that could never hold, so the refresh period was in practice unbounded. Initializing evidence during activation was deliberately not implemented, as stated when the finding was first answered, because the planned Filecoin FIP removes the `DataCap`-dependent evidence flow; that decision is unchanged.

[FIDL-fcd6bac3-M04] Stalled deals permanently strand provider capacity

Status: Resolved | Severity: Medium

Our perspective. A deal that cannot complete activation must have a terminal path that releases the provider reservation and closes the payment rail.

Current state. The implementation on main provides release paths for stalled deals, including accepted deals whose allocations expire. Such deals can be closed as `EXPIRED`, their payment rail can be terminated and the reserved provider capacity is returned. Additional economic hardening, such as a proposal deposit and allowances derived more closely from requested size and offer price, remains a product-design option rather than a prerequisite for closing this finding.

[FIDL-fcd6bac3-M05] Exact-match evidence gate cancels payment for terminated claims

Status: Accepted design | Severity: Medium

Our perspective. The exact-match gate reflects the current business rule: if a storage provider loses any part of the committed dataset, future payments stop. The team does not treat partial coverage as equivalent performance for the existing product.

Current state. No automatic reduction of committedBytes is planned for the current DataCap flow. A controlled repair or claim-replacement mechanism may be reconsidered in the post-DataCap architecture, where the evidence model may move or materially change. Until then, the all-or-nothing coverage rule is intentional and should be documented for providers and clients.

[FIDL-fcd6bac3-M06] Settlement gate does not validate evidence activeness

Status: Resolved | Severity: Medium

Our perspective. Inactive or stale evidence must defer payment without consuming the settlement window.

Current state. The implementation on main explicitly handles EvidenceResult.INACTIVE and stale evidence. validateDealSettlement reverts with the named custom error EvidenceTooStale when the adapter reports INACTIVE, or when the last completed evidence refresh precedes the settlement window by more than the eight-day grace period. The settlement window is therefore not consumed and the period can be settled after evidence is restored.

Post-audit remediation. The mechanism changed after this response was first written, so the description above supersedes the earlier one. The previous implementation returned a rejected decision with settleUpto set to fromEpoch; it now reverts instead. Both preserve the period for a later retry, but the revert makes the operational reason visible to the caller rather than surfacing as a generic FilecoinPay no-progress condition. This is the same change that closes L08, and the status is unchanged.

[FIDL-fcd6bac3-M07] Partial evidence refresh blocks settlement

Status: Resolved | Severity: Medium

Our perspective. A normal multi-batch refresh must not replace the last complete snapshot until the new sweep finishes.

Current state. The implementation on main accumulates in-progress bytes in pendingActiveClaimedBytes while retaining the last completed evidence snapshot as authoritative. A PARTIAL response is returned to the refresh caller, but settlement continues to read the last complete snapshot until the sweep completes.

[FIDL-fcd6bac3-M08] Unprovided evidence permanently blocks deal activation

Status: Resolved | Severity: Medium

Our perspective. The business rule still requires the full expected allocation set to be claimed before activation. The defect was the absence of a terminal recovery path, not the requirement for complete evidence.

Current state. The implementation on main allows an unrecoverable accepted deal to be closed as EXPIRED after the relevant allocation expiry, terminates its rail and releases the reserved provider capacity. The planned Filecoin FIP will eliminate the DataCap token entirely, so this DataCap-specific failure mode will no longer exist.

[FIDL-fcd6bac3-M09] Manifest hash poisoning blocks provider matching

Status: Post-audit resolved | Severity: Medium

Our perspective. A manifest assignment lock must not allow one client to exclude providers for another client's manifest. Organization-level scoping alone does not fully express the intended ownership boundary.

Current state. The manifest assignment is scoped by the client on main. SPRegistry stores assignments in `_clientManifestDealWithOrganization`, a mapping from client to manifest hash to organization, so a manifest hash occupied by one client is invisible to every other client's matching. PoRepMarket passes the deal client into every reservation and release path: `reserveProviderForDeal`, `reserveOfferForDeal`, `updatePendingReservation`, `releaseCapacity` and `releasePendingCapacity` all take the client explicitly, and the reserving entry points are restricted to `MARKET_ROLE` so only PoRepMarket can create an assignment. Cross-client poisoning is therefore no longer possible.

Post-audit remediation. The implementation differs from the approach described when the finding was first answered. Rather than deriving a composite assignment key, the existing mapping was restructured into a client-first nested mapping and the client was threaded through the whole reservation lifecycle, including the release paths, so an assignment is always cleared under the same client scope that created it. The public view was renamed from `isManifestAssignedToOrganization` to `isManifestAssignedToOrganizationAndClient`, which is a breaking ABI change for integrators. The organization dimension required by the matching model is retained.

4. Low-severity findings

[FIDL-fcd6bac3-L01] Provider matching can be steered by inflating a competitor pending load

Status: Planned for post-v2 | Severity: Low

Our perspective. The finding identifies a valid economic and ordering concern. Pending capacity is useful for preventing overbooking, but cost-free reservations can be used to influence the least-loaded component of matching.

Current state. The team is aware of the finding and is deliberately leaving the behaviour as it is for now. No economic resistance mechanism is implemented in the current release: the match load metric still reads `pendingBytes` and a reservation still costs only gas. The contract-level levers available today are too blunt to apply, because the project does not want to be that restrictive. Using `committedBytes` alone would avoid transient inflation but would equally ignore reservations that are legitimately in flight, and a hard anti-spam limit would constrain ordinary clients as much as an attacker. A durable answer is an economic mechanism rather than a contained contract change, so it is scheduled for a release after PoRep Market v2.

[FIDL-fcd6bac3-L02] Early termination path does not cap settlement at service end

Status: Post-audit resolved | Severity: Low

Our perspective. No termination record or settlement window should extend beyond the contracted service end.

Current state. The termination path on main cannot record a termination beyond the contracted service end. `_terminateDeal` reads the deal's `serviceEndEpoch` and reverts with `ServiceAlreadyEnded` when the current epoch is already past it, so `earlyTerminationEpoch` is by construction never greater than `serviceEndEpoch`. Settlement keeps an independent bound: `validateDealSettlement` rejects a window whose `fromEpoch` is past `serviceEndEpoch` and, on the non-terminated path, caps `effectiveToEpoch` at `serviceEndEpoch`.

Post-audit remediation. The implementation differs from the plan in one respect. Instead of clamping `earlyTerminationEpoch` to the service end when it is recorded, the contract rejects the termination outright once the service window has closed. A deal whose service has ended is finalized rather than terminated, so the clamped case has no legitimate caller. The defense-in-depth cap in the settlement path was kept.

[FIDL-fcd6bac3-L03] Settlement eligibility uses a point-in-time snapshot

Status: Post-audit resolved | Severity: Low

Our perspective. The production oracle aggregates observations for the full reporting period and publishes the resulting SLI values monthly. The attestation is therefore based on historical service data rather than a single instantaneous measurement. However, the on-chain interface does not encode the covered epoch interval, so that relationship is operational rather than enforced by the contract.

Current state. This is already handled on the oracle side. The production oracle aggregates observations across the full reporting period and publishes the resulting SLI values on a monthly cadence, and the finalization process was confirmed with the oracle operator, so the value read at settlement and at finalization reflects the whole reporting period rather than a single instantaneous measurement. No additional on-chain requirement for a final SLI update immediately before `finalizeDeal` proved necessary.

Post-audit remediation. The remediation is operational rather than on-chain, which is why the contracts are unchanged. The planned action was to confirm the finalization process with the oracle operator and to decide whether a final SLI update had to be required immediately before `finalizeDeal`; that confirmation was completed and the oracle's aggregation already satisfies the requirement. The on-chain interface still does not encode the covered epoch interval, so the relationship remains a property of the oracle rather than one enforced by the contract. Encoding explicit epoch windows in attestations and scoring stays available as longer-term hardening, but is not a prerequisite for closing this finding.

[FIDL-fcd6bac3-L04] No registry of consumed claims allows cross-deal reuse**Status: Post-audit resolved | Severity: Low**

Our perspective. Extending an existing claim within the same deal is a valid operational behavior and may occur periodically over a multi-year term. The prohibited behavior is counting the same claim as new evidence for a different deal or counting it multiple times for billing and capacity.

Current state. The fix is merged on main. `_registerAllocationOrClaimId` records the owning deal for every allocation and claim id in `_dealByAllocationOrClaimId` and reverts with `AllocationOrClaimIdAssignedToAnotherDeal` when an id already belongs to a different deal. Repeated evidence for the same deal is accepted but counted once: `_verifyAndRegisterClaimExtensions` adds a claim's size to `newEvidenceBytes` only the first time the id is registered, while the `DataCap` spend total still reflects the full extension, so legitimate periodic extensions of an existing claim continue to work without inflating deal accounting or provider capacity.

Post-audit remediation. The reconciliation flagged at the time was resolved by keeping the two mappings separate rather than merging them: `_dealByAllocationOrClaimId` carries claim ownership, and `_claimDealIds` routes an early-terminated claim to its deal so `_terminationRevisions` is incremented for that deal only. Once the `DataCap` token is retired, this issue will no longer exist.

[FIDL-fcd6bac3-L05] Capped settlement window blocks payment indefinitely**Status: Planned for post-v2 | Severity: Low**

Our perspective. The project recognizes the underfunded-payer scenario. `FilecoinPay` may offer a shorter funded interval, while `PoRepMarket` currently applies the monthly minimum to the offered window and can therefore reject the funded portion.

Current state. `FilecoinPay` coverage notifications and the planned collateral model improve detection and incentives but do not replace the contract fix, and the contract fix is not in this release. `validateDealSettlement` still derives the earliest permitted settlement end from `fromEpoch` plus `minTimeBetweenSettlementsInEpochs` rather than from the time elapsed since the previous settlement, so a shorter funded interval offered by `FilecoinPay` is still refused. The L08 change did alter how the refusal surfaces: it is now an explicit `SettlementTooEarly` revert naming the earliest permitted epoch instead of a silent lack of progress. A `lastSettledEpoch` field was added to `DealService` and is recorded on every accepted settlement, which is the state an elapsed-time rule would read, but the cadence check does not use it yet.

Planned action. The team is aware of the underfunded-payer scenario and is addressing it as part of a deposit-based collateral model rather than as an isolated change to the cadence rule, because detection, incentives and the settlement condition have to move together to avoid trading one failure mode for another. That is a larger feature than a contract patch and is scheduled for a release after `PoRep Market v2`.

[FIDL-fcd6bac3-L06] Capacity commitment ignores outstanding pending bytes

Status: Accepted design | Severity: Low

Our perspective. The principal trigger is an actual commitment larger than the deal's reservation. Preventing evidence above requestedSizeBytes removes that normal path, but the registry should still preserve committedBytes + pendingBytes <= availableBytes directly.

Current state. The behaviour is intentional. Allocation sizes are padded piece sizes, so an activation landing slightly above requestedSizeBytes is a normal outcome rather than an error, and the project does not want to be more restrictive than the tolerance it already grants. The overshoot is bounded on both sides: the DataCap adapter rejects evidence beyond requestedSizeBytes increased by the activation padding, which is 1000 basis points in the current deployment and capped at 2000 by MAX_DEAL_ACTIVATION_PADDING, and SPRegistry.updateAvailableSpace refuses to lower availableBytes below committedBytes plus pendingBytes. commitCapacity therefore validates committedBytes against availableBytes without re-checking the sum against the remaining pendingBytes.

Accepted risk. The accepted consequence is that committedBytes plus pendingBytes can exceed availableBytes by at most the activation padding, because the reservation released at activation is the size the deal requested while the amount recorded is the size actually claimed. A provider booked to its full advertised capacity may then need to raise availableBytes before a further deal on that provider can activate. The alternative was considered and rejected: refusing an activation that falls within the tolerance the protocol itself grants would penalise a deal that has already posted its DataCap and sealed its sectors, which is a worse outcome than asking the provider to restate its capacity. The team prefers to observe how the padding behaves in production before tightening the rule.

[FIDL-fcd6bac3-L07] Privileged role transfers are single-step

Status: Post-audit resolved | Severity: Low

Our perspective. The project accepts the recommendation. A privileged role should not be irrevocably moved until the receiving account proves control, especially where grantRole, revokeRole and renounceRole are intentionally disabled.

Current state. Privileged role management was centralized and made two-step on main. All protocol contracts route role checks through a single immutable AccessManager via the AccessControlledUpgradeable adapter, and AccessManager extends OpenZeppelin AccessControlDefaultAdminRules with a three-day INITIAL_DEFAULT_ADMIN_DELAY. Transferring the default admin therefore requires beginDefaultAdminTransfer, a three-day wait and an explicit acceptDefaultAdminTransfer from the receiving account, which proves control before the role moves, and the pending transfer can be cancelled in the meantime. Renouncing DEFAULT_ADMIN_ROLE is rejected.

Post-audit remediation. The change is broader than the two-step transfer originally planned. The audited pattern, `ValidatorFactory.setAdmin` and `setUpgraderRole` performing an atomic revoke-and-grant while `grantRole`, `revokeRole` and `renounceRole` were disabled, was removed entirely. The remaining roles, such as `UPGRADER_ROLE` and `ORACLE_ROLE`, are ordinary grant and revoke operations held by the delayed default admin; because role management is no longer disabled, a misassigned role is recoverable by the admin instead of permanent. Beacon upgrades owned by the manager are restricted to a dedicated `upgradeBeacon` operation.

[FIDL-fcd6bac3-L08] Deferred settlement reverts instead of reporting no progress

Status: Post-audit resolved | Severity: Low

Our perspective. Terminal conditions and temporary deferrals require different behavior. For ended or terminated periods, the cursor can advance because no further validated payment is expected. For a temporary condition, the caller should receive an explicit reason without accidentally consuming the period.

Current state. Terminal and temporary conditions are now distinguished on main. For terminal conditions the cursor may advance: a window starting past the service end returns `DEAL_ENDED` and a window starting past the termination epoch returns `DEAL_TERMINATED`, both with `settleUpto` set to the requested `toEpoch`. Temporary deferrals fail early with named custom errors instead of returning an unchanged cursor: `SettlementTooEarly` carries the requested and the earliest permitted end epoch, and `EvidenceTooStale` signals inactive or stale evidence. The operational reason is therefore preserved for the caller and no longer surfaces as a generic `FilecoinPayNoProgressInSettlement`.

Post-audit remediation. The two branches that returned a rejection with an unchanged cursor were replaced with reverts. The named errors are the ones described in the planned action, and `SettlementTooEarly` additionally carries parameters so the caller can compute when the settlement becomes eligible.

5. Informational recommendations

[FIDL-fcd6bac3-R01] Bound the deal activation padding

Status: Post-audit resolved | Severity: Informational

Our perspective. A 100% activation tolerance can disable the minimum-size check and is wider than required for sector-rounding differences.

Current state. `MAX_DEAL_ACTIVATION_PADDING` is reduced from 10000 to 2000 basis points on main, so the maximum tolerance is 20 percent and a 100 percent tolerance can no longer disable the minimum-size check. `initialize` assigns the padding explicitly, setting `_dealActivationPadding` to 1000 basis points, so a fresh deployment starts at a 10 percent tolerance rather than at an implicit zero. `setDealActivationPadding` continues to reject anything above the maximum with `DealActivationPaddingTooHigh`.

Post-audit remediation. Two changes were made. The padding is now assigned explicitly during initialization, at 1000 basis points, so a fresh deployment starts at a 10 percent tolerance rather than at an implicit zero. On top of that a hard ceiling was added, so the maximum deviation an administrator can configure is 20 percent; `setDealActivationPadding` rejects anything above it with `DealActivationPaddingTooHigh`. That ceiling is 2000 basis points rather than the 500 anticipated when the finding was first answered, because the wider value accommodates padded piece sizes, which is also why the same constant bounds posted evidence under M02. Zero remains a valid value, so a deployment that wants an exact-size rule can still configure one.

[FIDL-fcd6bac3-R02] Compare the transferred DataCap amount numerically

Status: Not applicable | Severity: Informational

Our perspective. The affected byte-level comparison existed only in `rescueDealAllocations`.

Current state. The entire `rescueDealAllocations` path was removed from the implementation on main, together with `RESCUE_ROLE` and the associated event. The problematic comparison no longer exists, so this recommendation is resolved by code removal rather than by introducing `BigInts.toUint256`.

[FIDL-fcd6bac3-R03] Use the storage accessor consistently

Status: Post-audit resolved | Severity: Informational

Our perspective. Deal existence checks and future invariants should be centralized rather than applied inconsistently across direct mapping reads.

Current state. Ordinary deal reads in `DataCapEvidenceAdapter` are routed through `_getStorageDeal` on main. At the time of the audit twelve direct mapping reads bypassed the accessor; two remain, and both are the intentional exceptions described here: the existence probe in `submitDataCapBatch` that decides whether the deal record must still be created, and `_registerDeal`, which writes the record before it exists.

Post-audit remediation. The approach follows the one described when the finding was answered. The two remaining raw reads sit at deal registration, where an accessor that assumes an existing record would be wrong.

[FIDL-fcd6bac3-R04] Reuse the allocation validation helper

Status: Not applicable | Severity: Informational

Our perspective. The duplicated validation was located in `rescueDealAllocations`.

Current state. `rescueDealAllocations` was removed from the implementation on main, eliminating the duplicate validation block. No further consolidation is required for this recommendation unless a replacement rescue flow is introduced in the future.

[FIDL-fcd6bac3-R05] Pin the transfer destination when posting DataCap**Status: Post-audit resolved | Severity: Informational**

Our perspective. The DataCap actor currently rejects an invalid destination, so this is defense in depth. Local validation nevertheless makes the contract's intent explicit and avoids relying exclusively on the downstream actor.

Current state. `submitDataCapBatch` validates the transfer destination on main. It compares `params.to.data` against the address derived from `VerifRegTypes.ActorID` and reverts with `InvalidTransferDestination` on any mismatch, so the contract no longer relies exclusively on the DataCap actor to reject a wrong destination.

Post-audit remediation. The destination is still supplied by the caller and validated locally rather than constructed internally, which is the fallback stated when the finding was answered; constructing it internally would have changed the external `TransferParams` integration.

[FIDL-fcd6bac3-R06] Validate batch size before mutating refresh state**Status: Post-audit resolved | Severity: Informational**

Our perspective. The existing revert rolls back earlier writes, so no incorrect state persists. Validating first is still clearer and avoids unnecessary gas before a predictable failure.

Current state. `refreshEvidenceStatus` decodes `evidenceData` and rejects `batchSize == 0` with `InvalidBatchSize` as its first two statements, ahead of the allocation-status check and of any read or mutation of the refresh state. At the time of the audit the reset block ran before that check. `submitEvidenceBatch` applies the same guard.

[FIDL-fcd6bac3-R07] Use boolean negation instead of comparison**Status: Post-audit resolved | Severity: Informational**

Our perspective. This is a style-only change with no behavioral impact.

Current state. No explicit boolean comparisons remain in `src.postingFinished` and the other boolean flags are tested with direct negation.

[FIDL-fcd6bac3-R08] Decode the count field with a matching width**Status: No change planned | Severity: Informational**

Our perspective. `readUInt64` correctly decodes every valid u32 CBOR value, and the decoded `success_count` value is discarded. The caller is the DataCap actor, so accepting a wider malformed value has no practical consequence in this path.

Current state. The project accepts the minor documentation argument but does not consider a code change necessary. The existing decoding is functionally correct and the field is not used for a security decision.

[FIDL-fcd6bac3-R09] Correct the stale bandwidth cap comment

Status: Post-audit resolved | Severity: Informational

Our perspective. Documentation must not imply a limit that is not enforced by either SLI validator.

Current state. The obsolete 64 Gbps statement was removed from the `bandwidthBytesPerSecond` field in `SharedTypes`. No bandwidth cap was introduced and neither SLI validator enforces one, so the documentation and the implementation now agree.

[FIDL-fcd6bac3-R10] Missing paused-state checks

Status: Post-audit resolved | Severity: Informational

Our perspective. Paused providers are already excluded during matching, but they should not be able to create or reactivate offers that cannot be selected. Offer deactivation must remain available while paused.

Current state. `createOffer` calls `_ensureProviderNotPaused`, and `setOfferActive` calls it only when activating, so a paused provider can neither create an offer nor reactivate one while offer deactivation remains available. Matching continues to skip paused providers, returning `OfferMatch.PROVIDER_PAUSED`.

Post-audit remediation. The optional part of the plan was not adopted: pausing a provider still leaves its offers in `_activeOfferIds`, and they are excluded from selection by the per-offer paused check in `_checkOfferEligibility` rather than by removal from the set.

[FIDL-fcd6bac3-R11] Paginate unbounded view functions

Status: Post-audit resolved | Severity: Informational

Our perspective. Unbounded view calls can eventually exceed node `eth_call` gas limits even when they are never used on a chain. Paginated alternatives are the established pattern in the current contracts.

Current state. The identified getters are paginated on main. `PoRepMarket.getDeals` and `getDealsForOrganizationByState` take `offset` and `limit` and return the total alongside the page, as do the new `getDealIds` and `getDealIdsByState`, with `getDealCount` exposing the count on its own. `SPRegistry.getProviders` follows the same pattern, and `PoRepMarketViewHelper` was moved onto the paginated APIs.

Post-audit remediation. One deviation from the plan: the unbounded signatures were replaced rather than retained as deprecated, so the change is breaking for integrators that call the previous `getDeals`, `getDealsForOrganizationByState` or `getProviders`.

[FIDL-fcd6bac3-R12] Remove dead code and unused declarations

Status: Post-audit resolved | Severity: Informational

Our perspective. Dead recovery paths and unreachable validation branches make the intended state machine harder to understand and maintain.

Current state. `_ensureAllocationSizeWithinTolerance`, `InvalidAllocationSizeForDealActivation` and `Validator.InvalidDealId` were removed from the implementation on main, and `activatePayment` has since been removed as well. The offer-specific reservation functions are no longer dead: `proposeDealWithSpecificOffer` calls `reserveOfferForDeal`, and `previewOfferForDeal` remains a useful read API. Two items were reviewed and are deliberately retained: the zero-amount branch in `_calculateAmountPerEpoch`, which is unreachable under the current pricing invariants but is a cheap guard should those invariants change, and the commented `MinerUtils.getOwner` block, which the code marks as temporarily commented out for coverage purposes. Neither is reachable in production and neither affects behaviour.

Planned action. No further action is planned. The removal of `activatePayment` is complete, and the two remaining items were reviewed and are retained deliberately.