XRPL Commons — Vote commons-vote-XLS65

Amendment Vote Position — XLS-0065 Single Asset Vault

Issued by XRPL Commons · Position Paper

Audited build rippled `release-3.1`, tag `3.1.3` (`46b241ace8`)

Specification XLS-0065 Single Asset Vault

Date 26 May 2026

Position: Not voting at this time

XRPL Commons is not casting a vote on the activation of `featureSingleAssetVault` in its current form on `rippled 3.1.3` at this time. This is neither a vote in favour nor a vote against the amendment.

This is not a judgment on the design of XLS-0065, which we consider sound, nor on the engineering effort behind the implementation. The reason we are not voting yet is one specific specification-to-code conformance gap that has survived the 3.1.1 to 3.1.3 maintenance cycle and creates a permanent, zero-cost griefing primitive against any deployed Vault.

Our position is bounded: we will revisit it once the conditions listed in §6 are met. The principal finding admits a one-line upstream fix.

1. Executive summary

The audit covered the seven Vault transactors, their ledger dependencies, and the new `fixCleanup3_1_3` amendment introduced in `rippled 3.1.3`. The overall structure remains defensive, with serious invariant coverage in `InvariantCheck::ValidVault`. No vulnerability of **Critical** severity was identified, but one **Medium-leaning-High** issue, confirmed by proof-of-concept, undermines the operational safety of every Vault deployed in production.

The audit was first conducted on `tag 3.1.1` and reverified on `tag 3.1.3`. The maintenance cycle between the two tags shipped `fixCleanup3_1_3`, which addresses internal Vault accounting in the clawback and share-denominated withdraw paths. It does not address the principal finding.

Severity	Count	Status at 3.1.3
Critical	0	—
High (economic, likely)	1	F2 — Still present. Share math unchanged in `View.cpp`.
Medium (confirmed)	1	F1 — Still present. `Payment.cpp` not modified between 3.1.1 and 3.1.3.
Medium (likely)	2	F4 — Still present. F5 — Materially mitigated by `fixCleanup3_1_3`.
Info / dismissed	4	Hardening items, no direct impact.

The principal finding is the absence of an enforcement that the specification itself states as intended behaviour. The XLS-0065 FAQ states that the pseudo-account *"cannot receive funds"*. The rippled implementation enforces this for XRP-direct payments but not for IOU rippling or MPT direct payments. The rest of the report unfolds around this asymmetry and its operational consequences.

2. Principal risk — F1 (confirmed by PoC, persists in 3.1.3)

Any holder of a Vault's underlying asset can submit a Payment transaction directly to the Vault's pseudo-account. Funds delivered this way are silently locked:

- they do not increment the Vault's internal accounting (`assetsTotal`, `assetsAvailable`),
- neither `VaultWithdraw` nor `VaultClawback` can recover them, because both operate on the internal accounting fields, not on the raw pseudo-account balance,
- `VaultDelete` becomes impossible, because the pseudo-account's trustline shows a non-zero balance and the trustline-removal step refuses to proceed. The Vault owner permanently loses the doubled XRP reserve.

The attack surface requires no privilege beyond holding the underlying asset. The attacker's own funds are also burnt, which makes this a self-destructive but always-available griefing primitive — a profile that becomes economically rational in any adversarial environment (protocol-level competition, targeted disruption of an institutional Vault, manipulation of secondary share markets).

The fix is a single-line change to lift the existing `isPseudoAccount` guard above the asset-type branching in the Payment transactor. The XRP path already enforces it; the IOU and MPT paths return before reaching it. The source code already contains a comment describing the intended invariant. Importantly, this fix did not ship in `fixCleanup3_1_3` despite the same release cycle addressing adjacent issues; the gap appears not to have been triaged.

3. Secondary risk — F2 (economic, persists in 3.1.3)

The deposit formula computes share issuance against gross `assetsTotal`. The withdraw formula uses `assetsTotal - lossUnrealized`. The two formulas are individually faithful to the specification, but their asymmetry creates a non-documented economic dynamic: whenever a Vault carries unrealised loss, a new depositor immediately subsidises every existing holder.

Latent loss can only be inscribed by Lending Protocol (XLS-0066) transactors. F2 is dormant in a standalone Vault deployment and becomes exploitable as soon as XLS-0066 is activated. The exploit is front-runnable on public impairment transactions: an observer can deposit just before impairment and route the loss to the next depositor.

We do not recommend resolving F2 by code change alone. The matter requires an explicit position from the working group: either align the deposit denominator with the withdraw denominator, or block deposits while `lossUnrealized > 0`, or formally document the dynamic in both the specification and user-facing tooling. The current state — silent loss for uninformed depositors — is neither principled nor user-safe.

4. Tertiary risks

F4 — Withdraw routing to pseudo-account. Status: persists in 3.1.3. The destination field of a `VaultWithdraw` accepts any account address, including the pseudo-account of another Vault. This reproduces the F1 fund-locking primitive through a different path. `fixCleanup3_1_3` does add a share-to-asset conversion in `VaultWithdraw::preclaim` that materially improves the `canWithdraw` accuracy, but it does not introduce the `isPseudoAccount(destination)` guard. The fix shape is symmetrical to F1.

F5 — Vault owner can burn other holders' shares. Status: materially mitigated by `fixCleanup3_1_3`. The implementation continues to allow the Vault owner to clawback shares when the Vault holds no assets. Under `fixCleanup3_1_3`, the owner is now required to burn the entire holding of the targeted account atomically (`tecLIMIT_EXCEEDED` otherwise), which substantially reduces the selectivity of the operation. The remaining concern is documentary: the specification continues to describe clawback as *"Issuer-Exclusive"* and does not mention the additional owner power. We do not consider F5 a vote blocker post-`fixCleanup3_1_3`, but it should be reconciled in the specification text.

5. The `fixCleanup3_1_3` amendment — observation

The `rippled 3.1.3` release introduces a separate amendment, `fixCleanup3_1_3`, with `Supported::yes` and `VoteBehavior::DefaultYes`. It includes Vault hardening that addresses:

- Zero-amount clawback now correctly clamps to `assetsAvailable` via `sharesToAssetsWithdraw` (previously could over-recover when an outstanding loan was present).
- Share-denominated `VaultWithdraw` now passes the equivalent asset amount through `canWithdraw`, where the pre-amendment path skipped the limit check entirely.
- Owner clawback on shares is constrained to burn the full balance of the targeted holder, removing the selective-burn pattern flagged in F5.

The existence of this amendment validates the necessity of post-feature corrective work and confirms that the engineering team has been actively triaging issues in the Vault implementation. XRPL Commons supports the activation of `fixCleanup3_1_3` jointly with `featureSingleAssetVault`. The amendment does not however cover F1, which remains the reason we are not casting a vote on the Vault amendment at this time.

6. Conditions for revisiting the position

XRPL Commons will revisit its position once the following conditions are jointly met.

1. The F1 patch is merged into upstream `develop`, with a regression test covering the three asset paths (XRP, IOU rippling, MPT direct).
2. The F4 patch is merged.

3. The XLS working group has issued a public position on F2 — either a code change, a specification amendment with explicit user warning, or both.
4. The XLS specification is amended to document the post-`fixCleanup3_1_3` owner clawback semantics (formerly F5).
5. A stable rippled release (`3.1.x` or `3.2.0`) containing at least the F1 and F4 fixes has been published, validated on testnet for at least two weeks, with no regression reported.
6. `fixCleanup3_1_3` is activated jointly with or before `featureSingleAssetVault`.

We assess that conditions 1, 2, 5, and 6 are within reach of a single release cycle. Conditions 3 and 4 require working-group decisions rather than additional engineering and can run in parallel.

7. Coupling with XLS-0066

The Lending Protocol amendment formally depends on `featureSingleAssetVault`. Activating XLS-0065 before resolving F1 exposes every Vault that will later be used as collateral source by a `LoanBroker`. Because we are not casting a vote on XLS-0065 at this time, we are likewise not casting a vote on XLS-0066 at this time. The Commons position on XLS-0066 is documented in a companion paper.

8. Engagement in the event of activation before resolution

If a majority of validators were to activate the amendment before the above conditions are met, XRPL Commons would:

- continue to refrain from casting a vote on `featureSingleAssetVault` until the corrective release is published,
- maintain its support for `fixCleanup3_1_3` on its own merits,
- issue a user advisory recommending that any party deploying Vaults wait for the corrective release,
- accompany the community with defensive guidance for the interim period — notably that XRP-asset Vaults are unaffected by F1 in its IOU/MPT form.

Contact — `romain@xrpl-commons.org`