

TEMPLAR · SECURITY REVIEW

# Templar

# Blend Contracts V2.

A security review of Blend Contracts V2.

| CLIENT  | PROJECT            | AUDITORS                                                                                 | AUDIT WINDOW                    | FINAL REPORT     |
|---------|--------------------|------------------------------------------------------------------------------------------|---------------------------------|------------------|
| Templar | Blend Contracts V2 | Jakub Zmyslowski,<br>Vlad Zotov,<br>Michael Lett,<br>Ferran Celades,<br>Said Abdul Malik | October 1 –<br>October 21, 2026 | October 21, 2026 |

§ 00

# Contents.

29 pages: overview, confidence ranking, scope, 15 findings, disclaimer, and about Guardian.

|      |                           |       |
|------|---------------------------|-------|
| § 01 | Overview                  | P. 03 |
| § 02 | Confidence Ranking        | P. 04 |
| § 03 | Scope & Classifications   | P. 05 |
|      | Files in Scope            | P. 06 |
| § 04 | Findings                  | P. 08 |
|      | Main Review · 15 findings | P. 09 |
| § 05 | Disclaimer                | P. 28 |
| § 06 | About Guardian            | P. 29 |

§ 01

# Overview.

Guardian reviewed Blend Contracts V2 across 2 review rounds. The final report includes 5 Low, 10 Informational findings.

|                                                                              |                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AUDIT FIRM                                                                   | REPOSITORY                                                                                                                                                                                                                                                                                                                                                                  |
| Guardian                                                                     | <a href="https://github.com/Templar-Protocol/blend-contracts-v2">https://github.com/Templar-Protocol/blend-contracts-v2</a> , <a href="https://github.com/Templar-Protocol/precept/pull/16/changes">https://github.com/Templar-Protocol/precept/pull/16/changes</a> , <a href="https://github.com/Templar-Protocol/precept">https://github.com/Templar-Protocol/precept</a> |
| CLIENT                                                                       | METHODOLOGY                                                                                                                                                                                                                                                                                                                                                                 |
| Templar                                                                      | Smart contract · Manual · Main Review · Remediation Review                                                                                                                                                                                                                                                                                                                  |
| PROJECT                                                                      | MAIN REVIEW COMMIT                                                                                                                                                                                                                                                                                                                                                          |
| Blend Contracts V2                                                           | 7a09b0e3d228                                                                                                                                                                                                                                                                                                                                                                |
| PREPARED BY                                                                  | REMEDIATION REVIEW COMMIT                                                                                                                                                                                                                                                                                                                                                   |
| Jakub Zmyslowski, Vlad Zotov, Michael Lett, Ferran Celades, Said Abdul Malik | b385dfb5cf18                                                                                                                                                                                                                                                                                                                                                                |
| AUDIT WINDOW                                                                 |                                                                                                                                                                                                                                                                                                                                                                             |
| October 1 – October 21, 2026                                                 |                                                                                                                                                                                                                                                                                                                                                                             |
| FINAL REPORT                                                                 |                                                                                                                                                                                                                                                                                                                                                                             |
| October 21, 2026                                                             |                                                                                                                                                                                                                                                                                                                                                                             |

## 1.1 Confidence Ranking

5 / 5

Very High Confidence

Guardian assigns a Confidence Ranking of 5 out of 5, combining finding severity, remediation outcome, and the reviewer’s assessment of latent risk. The full scale is on the next page.

|                     |                |                     |                 |                      |
|---------------------|----------------|---------------------|-----------------|----------------------|
| 1                   | 2              | 3                   | 4               | 5                    |
| VERY LOW CONFIDENCE | LOW CONFIDENCE | MODERATE CONFIDENCE | HIGH CONFIDENCE | VERY HIGH CONFIDENCE |

## 1.2 Findings at a Glance

| SEVERITY                  | TOTAL | PENDING | ACK | PARTIAL | RESOLVED |
|---------------------------|-------|---------|-----|---------|----------|
| <div></div> Critical      | 0     | 0       | 0   | 0       | 0        |
| <div></div> High          | 0     | 0       | 0   | 0       | 0        |
| <div></div> Medium        | 0     | 0       | 0   | 0       | 0        |
| <div></div> Low           | 5     | 0       | 1   | 0       | 4        |
| <div></div> Informational | 10    | 0       | 0   | 0       | 10       |
| All severities            | 15    | 0       | 1   | 0       | 14       |

## § 02

# Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

| CONFIDENCE RANKING                               | DEFINITION AND RECOMMENDATION                                                                                                                                                                                               | RISK PROFILE                                                                              |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <b>5 Very High Confidence</b><br><b>ASSIGNED</b> | Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: code is highly secure at time of audit. | 0 High/Critical findings and few Low/Medium severity findings.                            |
| <b>4 High Confidence</b>                         | Code is clean, well-structured, and adheres to best practices. Design patterns are sound and test coverage is strong. Recommendation: suitable for deployment after remediations; consider periodic review with changes.    | 0-1 High/Critical findings per engagement week and little to no Medium severity findings. |
| <b>3 Moderate Confidence</b>                     | Code is functional but shows gaps in defensive programming or test coverage. Recommendation: remediate the findings and re-review the changes before deployment.                                                            | Some High/Critical findings, with various Medium severity findings.                       |
| <b>2 Low Confidence</b>                          | Recurring issues suggest systemic gaps rather than isolated mistakes. Recommendation: remediate, then commission a follow-up review at a frozen commit.                                                                     | Over 6 High and Critical findings across the engagement.                                  |
| <b>1 Very Low Confidence</b>                     | The volume and severity of findings indicate the codebase is not ready for production. Recommendation: substantial rework, then a full re-review.                                                                           | Over 12 High and Critical findings across the engagement.                                 |

## § 03

# Scope & Classifications.

Guardian reviewed Blend Contracts V2 in <https://github.com/Templar-Protocol/blend-contracts-v2>, <https://github.com/Templar-Protocol/precept/pull/16/changes>, <https://github.com/Templar-Protocol/precept>. The scope includes 1,407 source lines across 46 files.

## 3.1 Severity Matrix

|            | IMP · HIGH | IMP · MED | IMP · LOW |
|------------|------------|-----------|-----------|
| LIK · HIGH | Critical   | High      | Medium    |
| LIK · MED  | High       | Medium    | Low       |
| LIK · LOW  | Medium     | Low       | Low       |

### IMPACT · HIGH

Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.

### IMPACT · MEDIUM

Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.

### IMPACT · LOW

Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.

### LIKELIHOOD · HIGH

Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.

### LIKELIHOOD · MEDIUM

Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.

### LIKELIHOOD · LOW

Unlikely to ever occur in production.

## Files in Scope 1/2.

| FILE                                           | NSLOC | TOTAL |
|------------------------------------------------|-------|-------|
| <b>MAIN REVIEW</b>                             |       |       |
| backstop/src/backstop/deposit.rs               | 0     | 298   |
| backstop/src/backstop/mod.rs                   | 1     | 19    |
| backstop/src/backstop/user.rs                  | 1     | 774   |
| backstop/src/backstop/withdrawal.rs            | 8     | 792   |
| backstop/src/constants.rs                      | 1     | 18    |
| backstop/src/contract.rs                       | 12    | 319   |
| backstop/src/dependencies/mod.rs               | 2     | 13    |
| backstop/src/dependencies/pool.rs              | 13    | 22    |
| backstop/src/emissions/claim.rs                | 29    | 957   |
| backstop/src/emissions/distributor.rs          | 11    | 746   |
| backstop/src/emissions/manager.rs              | 60    | 2,305 |
| backstop/src/emissions/mod.rs                  | 1     | 8     |
| backstop/src/errors.rs                         | 1     | 32    |
| backstop/src/events.rs                         | 0     | 174   |
| backstop/src/lib.rs                            | 2     | 24    |
| backstop/src/storage.rs                        | 23    | 518   |
| pool-factory/src/pool_factory.rs               | 1     | 122   |
| pool/src/auctions/auction.rs                   | 79    | 2,923 |
| pool/src/auctions/backstop_interest_auction.rs | 5     | 1,325 |
| pool/src/auctions/bad_debt_auction.rs          | 13    | 2,357 |
| pool/src/auctions/user_liquidation_auction.rs  | 6     | 3,585 |
| pool/src/constants.rs                          | 1     | 16    |
| pool/src/contract.rs                           | 40    | 584   |
| pool/src/emissions/distributor.rs              | 0     | 1,667 |
| pool/src/emissions/manager.rs                  | 4     | 656   |
| pool/src/errors.rs                             | 1     | 55    |
| pool/src/events.rs                             | 17    | 398   |
| pool/src/lib.rs                                | 3     | 32    |
| pool/src/math.rs                               | 14    | 39    |
| pool/src/pool/actions.rs                       | 72    | 1,920 |
| pool/src/pool/bad_debt.rs                      | 136   | 1,534 |
| pool/src/pool/config.rs                        | 64    | 1,608 |
| pool/src/pool/gulp.rs                          | 18    | 483   |
| pool/src/pool/health_factor.rs                 | 0     | 359   |
| pool/src/pool/interest.rs                      | 43    | 455   |

## Files in Scope 2/2.

| FILE                                        | NSLOC | TOTAL |
|---------------------------------------------|-------|-------|
| pool/src/pool/mod.rs                        | 1     | 38    |
| pool/src/pool/pool.rs                       | 17    | 1,051 |
| pool/src/pool/reserve.rs                    | 86    | 953   |
| pool/src/pool/status.rs                     | 20    | 1,188 |
| pool/src/pool/submit.rs                     | 23    | 2,674 |
| pool/src/pool/user.rs                       | 23    | 1,252 |
| pool/src/storage.rs                         | 29    | 661   |
| shared/backstop_threshold.rs                | 11    | 15    |
| contracts/pool-oracle-adapter/src/lib.rs    | 192   | 226   |
| contracts/pool-oracle-adapter/src/parent.rs | 7     | 8     |
| <b>REMEDIATION REVIEW</b>                   |       |       |
| pool-factory/src/pool_factory.rs            | 1     | 119   |
| pool/src/auctions/auction.rs                | 16    | 2,958 |
| pool/src/contract.rs                        | 0     | 598   |
| pool/src/emissions/distributor.rs           | 1     | 1,667 |
| pool/src/events.rs                          | 3     | 401   |
| pool/src/lib.rs                             | 1     | 33    |
| pool/src/pool/actions.rs                    | 80    | 2,475 |
| pool/src/pool/bad_debt.rs                   | 10    | 1,626 |
| pool/src/pool/config.rs                     | 3     | 1,709 |
| pool/src/pool/interest.rs                   | 0     | 463   |
| pool/src/pool/mod.rs                        | 2     | 38    |
| pool/src/pool/pool.rs                       | 0     | 1,052 |
| pool/src/pool/reserve.rs                    | 11    | 965   |
| pool/src/pool/status.rs                     | 21    | 1,271 |
| pool/src/pool/submit.rs                     | 46    | 2,722 |
| pool/src/pool/user.rs                       | 39    | 1,293 |
| pool/src/storage.rs                         | 79    | 733   |
| contracts/pool-oracle-adapter/src/lib.rs    | 3     | 231   |

## § 04

# Findings.

Grouped by round. Visible IDs reset by severity within each round, so a finding is named the same way in the report as it was in the channel it was written in. Status is where each finding ended up; its own page says how.

| ID                 | TITLE                                                                                              | CATEGORY              | SEVERITY      | STATUS       | PAGE  |
|--------------------|----------------------------------------------------------------------------------------------------|-----------------------|---------------|--------------|-------|
| <b>MAIN REVIEW</b> |                                                                                                    |                       |               |              |       |
| L-01               | Oracle failures block liquidation settlement                                                       | Oracle                | LOW           | RESOLVED     | p. 9  |
| L-02               | A borrower can block liquidation by adding dust collateral in a reserve whose price is unavailable | Oracle                | LOW           | RESOLVED     | p. 10 |
| L-03               | Admin proposal TTL can be extended                                                                 | Access Control        | LOW           | RESOLVED     | p. 12 |
| L-04               | Withdrawing with i128::MAX reverts after a default                                                 | Business logic errors | LOW           | RESOLVED     | p. 13 |
| L-05               | Adapter timestamp flooring defeats the pool's future-dated price rejection                         | Oracle                | LOW           | ACKNOWLEDGED | p. 15 |
| I-01               | Stale ABI documentation advertises unsupported bad-debt and interest auction creation              | Documentation         | INFORMATIONAL | RESOLVED     | p. 17 |
| I-02               | Positive balance changes remain unclaimable                                                        | Documentation         | INFORMATIONAL | RESOLVED     | p. 18 |
| I-03               | Freeze docs imply the pool can be reopened                                                         | Documentation         | INFORMATIONAL | RESOLVED     | p. 20 |
| I-04               | Bad debt docs misstate loss allocation                                                             | Documentation         | INFORMATIONAL | RESOLVED     | p. 21 |
| I-05               | Submit docs omit auction fill restrictions                                                         | Documentation         | INFORMATIONAL | RESOLVED     | p. 22 |
| I-06               | Activation accepts an incompatible oracle                                                          | Configuration         | INFORMATIONAL | RESOLVED     | p. 23 |
| I-07               | Factory accepts undeployable take rates                                                            | Validation            | INFORMATIONAL | RESOLVED     | p. 24 |
| I-08               | plan_repay comment says the refund can be negative                                                 | Documentation         | INFORMATIONAL | RESOLVED     | p. 25 |
| I-09               | del_auction docs say only unfilled auctions become stale                                           | Documentation         | INFORMATIONAL | RESOLVED     | p. 26 |
| I-10               | Stale comment in the pool factory                                                                  | Documentation         | INFORMATIONAL | RESOLVED     | p. 27 |

LOW **L-01**

RESOLVED

# Oracle failures block liquidation settlement

|          |                                                                                                                                                                                                                                                                                             |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Oracle                                                                                                                                                                                                                                                                                      |
| LOCATION | <a href="https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/bad_debt.rs#L53-L64">https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/bad_debt.rs#L53-L64</a> |
| REVIEW   | Main Review                                                                                                                                                                                                                                                                                 |

## — DESCRIPTION

When the borrower still has liabilities, `check_and_handle_user_bad_debt` calculates the borrower's full position value to check whether any collateral value remains. This calculation also fetches prices for outstanding debt assets, even when the borrower has no collateral. Those prices are unnecessary for using same reserve supply to repay debt or defaulting the remaining debt.

This affects liquidation completion because `fill_user_liq_auction` calls the helper on every full fill, after transferring the auction positions. If the borrower still has debt and a required oracle call fails or returns unusable prices, the entire transaction reverts. The position transfers are rolled back, and execution never reaches auction deletion. An already created auction therefore remains open despite the attempted full fill.

Standalone bad-debt settlement uses the same helper and encounters the same dependency.

An oracle outage can block affected full liquidation fills and prevent settlement of debt with no remaining collateral, delaying loss recognition and leaving debt outstanding.

## — RECOMMENDATION

Consider skipping oracle valuation when the borrower has no positive collateral balances. When collateral remains, calculate only its raw value without fetching debt asset prices.

## — RESOLUTION

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

LOW **L-02**

RESOLVED

## A borrower can block liquidation by adding dust collateral in a reserve whose price is unavailable

|          |                                                                                                                                                                                                                                                                                               |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Oracle                                                                                                                                                                                                                                                                                        |
| LOCATION | <a href="https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/actions.rs#L365-L382">https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/actions.rs#L365-L382</a> |
| REVIEW   | Main Review                                                                                                                                                                                                                                                                                   |

### — DESCRIPTION

`apply_supply_collateral` loads no price and runs no health check. `calculate_from_positions` loads a price for every reserve in which the user holds collateral or debt. `load_price` reverts with `InvalidPrice` (#1210) when the oracle returns no price, a price that is not positive, a price older than 24 hours, or a timestamp in the future. `create_user_liq_auction_data` values the whole position before it reads the requested lot and bid.

While one listed reserve's price is unavailable, a borrower with debt can still supply a few stroops of collateral in that reserve. Every `new_auction` against the borrower then reverts with #1210 until the price returns, whatever lot, bid or percentage is requested. The shortfall can grow during the outage, and can fall on the debt reserve's lenders if the debt is later defaulted.

Borrowers who already held collateral in that reserve before the outage are blocked the same way. This finding is about a borrower adding the dependency after the outage has started, to keep an unhealthy position out of liquidation. It concerns auction creation, not the oracle calls made when a liquidation settles, and it lasts as long as the outage.

### — RECOMMENDATION

In `apply_supply_collateral`, load the reserve's price when the user has liabilities, so the call reverts while that price is unusable. Do not value unpriced collateral at zero when creating auctions, because that would make healthy positions liquidatable during an outage.

### — RESOLUTION

Status: Resolved.

---

**L -** A borrower can block liquidation by adding dust collateral in a reserve **CONTINUED**  
**02** whose price is unavailable

---

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

LOW **L-03**

RESOLVED

# Admin proposal TTL can be extended

CATEGORY Access Control

LOCATION pool/src/storage.rs:239-247; pool/src/contract.rs:377-389

REVIEW Main Review

## DESCRIPTION

The pool stores the proposed administrator in temporary storage and assigns the entry a TTL of 10 days:

```
e.storage().temporary().extend_ttl(
    &Symbol::new(e, PROPOSED_ADMIN_KEY),
    10 * ONE_DAY_LEDGERS,
    10 * ONE_DAY_LEDGERS,
);
```

The `accept_admin` function does not enforce an explicit expiration ledger or timestamp; it only checks whether the temporary entry still exists. Any account can include the entry in a transaction's read-only footprint and extend its lifetime using `ExtendFootprintTTL0p`, whose behavior is described in Stellar's [state-archival documentation](https://developers.stellar.org/docs/learn/fundamentals/contract-development/storage/state-archival#extendfootprintttl0p). Stellar's [storage guidance](https://developers.stellar.org/docs/build/guides/storage/choosing-the-right-storage#temporary-storage) also warns that entry expiration is not a safe security boundary because anyone can extend the TTL.

Consequently, if the 10-day period is intended as an acceptance deadline, the proposed administrator can keep the entry alive and accept the role after that period. The stale proposal remains valid until the current administrator explicitly replaces it.

## RECOMMENDATION

If the proposal must expire after ten days, store an explicit expiration timestamp or ledger and enforce it in `accept_admin` instead of relying on the storage entry's TTL.

## RESOLUTION

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/15>

LOW **L-04**

RESOLVED

# Withdrawing with `i128::MAX` reverts after a default

CATEGORY Business logic errors

LOCATION <https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/actions.rs#L315-L328>; <https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/reserve.rs#L230-L232>

REVIEW Main Review

## DESCRIPTION

`plan_withdraw` accepts a request larger than the position. If the b-tokens needed exceed the user's balance, it burns the whole position and pays out its full value. The b-token conversion runs before this cap:

```
let mut to_burn = data.to_b_token_up(math, amount);
let mut tokens_out = amount;
if to_burn > cur_b_tokens {
    to_burn = cur_b_tokens;
    tokens_out = data.to_asset_from_b_token(math, cur_b_tokens);
}
```

`to_b_token_up` computes `ceil(amount * 1e12 / b_rate)`. When the product overflows `i128`, `fixed_mul_ceil` retries in `I256` and traps if the result does not fit back into `i128`. With `amount = i128::MAX`, the result only fits while `b_rate >= 1e12`.

`b_rate` starts at `1e12` and grows with interest, but every default lowers it through `default_b_rate`. Once total defaults exceed the interest accrued so far, `b_rate` drops below `1e12`. From then on, every `Withdraw` or `WithdrawCollateral` request for `i128::MAX` in that reserve reverts, until interest brings `b_rate` back above `1e12`.

Users, vaults and scripts that close a position by passing `i128::MAX` cannot withdraw right after a loss event, when they are most likely to exit.

## RECOMMENDATION

Check the request against the position value before converting it to b-tokens:

```
let position_value = data.to_asset_from_b_token(math, cur_b_tokens);
if data.b_rate > 0 && amount > position_value {
    return (position_value, cur_b_tokens);
}
(amount, data.to_b_token_up(math, amount))
```

**L-04** Withdrawing with i128::MAX reverts after a default

CONTINUED

This returns the same result as the current code for every request that does not overflow, and removes the revert for oversized requests.

**— RESOLUTION**

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/15>

LOW **L-05**

ACKNOWLEDGED

# Adapter timestamp flooring defeats the pool's future-dated price rejection

|          |                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Oracle                                                                                                                                                                                                                                                                                                                                                                                               |
| LOCATION | contracts/pool-oracle-adapter/src/lib.rs:211-223 (precept PR #16, caeab1e4c84007aba4711e6b3b59f9ca1104fc4b); <a href="https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/pool.rs#L128-L134">https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/pool.rs#L128-L134</a> |
| REVIEW   | Main Review                                                                                                                                                                                                                                                                                                                                                                                          |

## — DESCRIPTION

The pool reads every price through the SEP-40 adapter, which forwards to the Templar proxy oracle (the parent). `load_price` rejects a price whose timestamp is ahead of ledger time, which is what enforces ADR-0008's requirement that the pool never prices against a future-dated observation. Before the pool sees it, the adapter's `lastprice` rounds the parent's timestamp down to a multiple of the configured resolution, and applies no future-dated check of its own:

```
fn bucket_timestamp(timestamp: u64, resolution: u32) -> u64 {
    timestamp - timestamp % u64::from(resolution)
}
```

A timestamp that is ahead of ledger time but inside the current bucket rounds down to at or below the current time, so the pool's check never fires.

The parent accepts a source sample published up to `max_clock_drift_secs` ahead of ledger time. That setting is mandatory and capped at one hour. While such a sample is the newest history entry, a later sample with an equal or lower timestamp does not replace it, the parent reads its age as zero for as long as it is ahead of ledger time, and `refresh`, which anyone can call, keeps marking the cache current. The pool keeps pricing against that one sample as the market moves.

The pool can therefore act on a price up to an hour old. Take a borrower holding 12.5 WETH against 19,200 USD of debt. At a true price of 2,060 USD the collateral is worth 25,750 USD and the position is solvent, with a health factor of 1.02. If the parent freezes on a 3% dip to 2,000 USD, the pool liquidates that borrower and about 1,280 USD, roughly 5% of the position, moves to the liquidator. At the true price the same call is rejected as `InvalidLiquidation`. The same frozen price also lets 10,000 USD of own-reserve supply leave an underwater position that the true price blocks with `InvalidHf`. Both revert with `InvalidPrice` if the parent's timestamp reaches the pool unmodified.

---

**L -** Adapter timestamp flooring defeats the pool's future-dated price  
**05** rejection

---

**CONTINUED**

— **RECOMMENDATION**

In the parent, let an observation inside the freshness window replace a history head that is ahead of ledger time, so the head cannot outlast the clock and the age gate stops reading it as zero seconds old. Passing the parent's timestamp through unfloored in `lastprice` would also restore the pool's check, but on its own it blocks pricing for that asset until ledger time catches up.

— **RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-01

RESOLVED

# Stale ABI documentation advertises unsupported bad-debt and interest auction creation

CATEGORY Documentation

LOCATION packages/blend-contracts-v2/pool/src/auctions/auction.rs:create\_auction:L84-L94

REVIEW Main Review

## DESCRIPTION

The fork intentionally rejects auction types 1 and 2 before their constructors, as required by ADR-0008, but the public `new_auction` ABI comments still describe all three types as supported and the unreachable constructors/match arms remain in the shipped code. Calls receive the mandated `BadRequest` and no state is changed.

The stale ABI and retained dead paths can mislead integrators and increase maintenance/review risk only.

## RECOMMENDATION

Document at the ABI boundary that only `UserLiquidation` (type 0) is supported, or remove/mark unsupported enum arms and constructors while preserving the intentional rejection. Update `new_auction/get_auction/del_auction` comments consistently.

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/SentryRepo-Blend-Contracts-V2--Round-1-01a0fb6db9b571faa3ca02770aeef691/blob/refinery/poc-01a0fbb8-b552-7a2a-8b52-b3701d288007/test/refinery-poc/stale-abi-documentation-advertises-unsupported-bad-debt-and--f8184dc3.rs>

## RESOLUTION

Status: Resolved.

**Templar Team:** Resolved in <https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

## INFORMATIONAL I-02

RESOLVED

# Positive balance changes remain unclaimable

CATEGORY Documentation

LOCATION pool/src/pool/gulp.rs:7-30; pool/src/contract.rs:197-208

REVIEW Main Review

## DESCRIPTION

The scoped diff replaces surplus accounting in `execute_gulp` with settlement of pool-owned orphan shares. Previously, the function compared the pool's actual token balance with its recorded balance:

```
let reserve_token_balance =  
    reserve.total_supply(e) + reserve.data.backstop_credit  
    - reserve.total_liabilities(e);  
let token_balance_delta = pool_token_balance - reserve_token_balance;
```

Positive differences were allocated to supplier claims and backstop credit. The replacement only burns orphan supply shares once reserve debt is zero and returns `0`. It does not reconcile the underlying token balance or allocate surplus.

Direct token transfers and positive balance adjustments therefore do not increase anyone's withdrawable claim.

Ignoring surplus is intentional and enforced by the fork's tests. However, the public interface documentation still promises recovery of unaccounted tokens, including positive rebases.

Surplus remains unclaimable. If positive rebasing assets are supported, their additional value is not allocated to suppliers or the backstop.

## RECOMMENDATION

Document the surplus policy and update the interface documentation to match the implemented behavior. If positive balance gains should remain recoverable, introduce a separate reconciliation function that allocates verified surplus to the intended beneficiaries while preserving orphan-settlement semantics.

## RESOLUTION

Status: Resolved.

**I-02** Positive balance changes remain unclaimable

CONTINUED

**Templar Team:** Resolved in <https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

## INFORMATIONAL I-03

RESOLVED

# Freeze docs imply the pool can be reopened

CATEGORY Documentation  
LOCATION pool/src/contract.rs:178-195  
REVIEW Main Review

## DESCRIPTION

The public interface documentation says the admin can update a frozen pool's status through `set_status`. However, the fork makes status 4 irreversible. Setting status 4 again is allowed, but every attempt to leave it reverts:

```
if pool_config.status == 4 {  
    panic_with_error!(e, PoolError::StatusNotAllowed);  
}
```

Operators relying on these docs may treat the freeze as a temporary pause, although borrowing and new deposits cannot resume through a status change.

## RECOMMENDATION

Update the interface documentation to state that status 4 is permanent and that even the admin cannot reactivate the pool after freezing it.

## RESOLUTION

Status: Resolved.

**Templar Team:** <https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

## INFORMATIONAL I-04

RESOLVED

## Bad debt docs misstate loss allocation

|          |                              |
|----------|------------------------------|
| CATEGORY | Documentation                |
| LOCATION | pool/src/contract.rs:310-313 |
| REVIEW   | Main Review                  |

### — DESCRIPTION

The public documentation for `bad_debt` still states that the backstop takes over a borrower's bad debt. It also describes passing the backstop address to check its health and default its remaining debt.

The fork instead rejects calls targeting the backstop. When a borrower has no remaining collateral value, it uses their ordinary supply in the same reserve to repay debt, then writes off the remainder against suppliers.

The implementation is intentional, but the interface documentation describes protection the code does not provide. Integrators and lenders may misunderstand who absorbs losses, while keepers may attempt unsupported backstop settlement calls.

### — RECOMMENDATION

Update the docs to describe the actual settlement process and clearly state that backstop settlement calls are unsupported.

### — RESOLUTION

Status: Resolved.

**Templar Team:** <https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

## INFORMATIONAL I-05

RESOLVED

# Submit docs omit auction fill restrictions

|          |                                                                         |
|----------|-------------------------------------------------------------------------|
| CATEGORY | Documentation                                                           |
| LOCATION | packages/blend-contracts-v2/pool/src/auctions/auction.rs:fill:L151-L155 |
| REVIEW   | Main Review                                                             |

## — DESCRIPTION

Request types 7 and 8 are retained in the public RequestType ABI and dispatched to fill, but the fork intentionally rejects both before reading auction storage, as required by ADR-0008. The corresponding fill implementations remain unreachable and submit documentation does not warn callers that these request types are unsupported.

Because creation of these auction types is also intentionally blocked and direct residual-debt handling replaces them, no valid auction or required keeper path is stranded. The only effect is integrator confusion and dead-code maintenance risk.

## — RECOMMENDATION

Keep the atomic rejection required by ADR-0008, but document request types 7/8 as unsupported (or remove/mark them at the ABI boundary) and remove or clearly annotate unreachable fill implementations.

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/SentryRepo-Blend-Contracts-V2--Round-1-01a0fb6db9b571faa3ca02770aeef691/blob/refinery/poc-01a0fbb8-b552-7a2a-8b52-b3701d288007/test/refinery-poc/stale-submit-abi-advertises-bad-debt-and-interest-auction-fi-680053f1.rs>

## — RESOLUTION

Status: Resolved.

**Templar Team:** <https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

## INFORMATIONAL I-06

RESOLVED

# Activation accepts an incompatible oracle

|          |                                                      |
|----------|------------------------------------------------------|
| CATEGORY | Configuration                                        |
| LOCATION | pool/src/pool/config.rs:16; pool/src/pool/pool.rs:99 |
| REVIEW   | Main Review                                          |

## DESCRIPTION

Pool initialization stores the supplied oracle without validating its decimal precision.

The pool only enforces the required seven-decimal precision when a health-dependent operation later loads the oracle metadata:

```
let decimals = oracle_client.decimals();
if decimals != 7 {
    panic_with_error!(e, PoolError::InvalidPrice);
}
```

Neither the administrative status transition nor the permissionless activation process performs this validation. Supply and collateral-supply operations also do not access the oracle. Consequently, a pool configured with an incompatible oracle can leave setup, become active, and accept deposits before the configuration error is detected.

The oracle address cannot be changed after deployment because no setter exists and the retained pool-update entry point always rejects. This can expose depositors to an unusable lending market and force an operationally expensive pool replacement after funds have entered the pool.

## RECOMMENDATION

Consider validating that the oracle uses exactly seven decimals before permitting the pool to leave setup status.

## RESOLUTION

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/15>

## INFORMATIONAL I-07

RESOLVED

# Factory accepts undeployable take rates

CATEGORY Validation

LOCATION pool-factory/src/pool\_factory.rs:75-78; pool/src/pool/config.rs:16-29

REVIEW Main Review

## DESCRIPTION

The factory accepts any `backstop_take_rate` below 100%:

```
if backstop_take_rate >= SCALAR_7 {  
    panic_with_error!(&e, PoolFactoryError::InvalidPoolInitArgs);  
}
```

It then forwards the value to the pool constructor. However, pool initialization only permits a zero rate:

```
if *bstop_rate != 0 {  
    panic_with_error!(e, PoolError::InvalidPoolConfigArgs);  
}
```

Consequently, values from `1` through `SCALAR_7 - 1` pass the factory's validation but cause pool deployment to revert during construction.

Because deployment is atomic, this does not create a malformed pool and its impact is limited to failed deployments, wasted transaction fees, and misleading factory validation.

## RECOMMENDATION

Consider enforcing `backstop_take_rate` to equal zero in the factory and updating its interface documentation to reflect the pool's restriction.

## RESOLUTION

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/15>

## INFORMATIONAL I-08

RESOLVED

# plan\_repay comment says the refund can be negative

CATEGORY Documentation

LOCATION <https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/pool/actions.rs#L431-L436>

REVIEW Main Review

## DESCRIPTION

The doc comment on `plan_repay` says the refund "may be negative on the capped branch; production re-checks it is nonnegative before transferring". The capped branch cannot produce a negative refund:

- The branch runs when `floor(amount * 1e12 / d_rate) > cur_d_tokens`, so `amount * 1e12 >= (cur_d_tokens + 1) * d_rate`.
- `d_rate` starts at `1e12` and only grows, so `amount >= cur_d_tokens * d_rate / 1e12 + 1`.
- The amount charged is `ceil(cur_d_tokens * d_rate / 1e12)`, which is less than `cur_d_tokens * d_rate / 1e12 + 1`.

The refund on the capped branch is therefore always at least 1. The comment describes a state that cannot happen and can mislead anyone reviewing repay rounding.

## RECOMMENDATION

Update the comment to say the refund is always positive on the capped branch. The `require_nonnegative` check can stay.

## RESOLUTION

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/15>

## INFORMATIONAL I-09

RESOLVED

# del\_auction docs say only unfilled auctions become stale

CATEGORY Documentation

LOCATION <https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/contract.rs#L297-L299>; <https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/auctions/auction.rs#L100-L112>; <https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool/src/auctions/auction.rs#L208-L213>

REVIEW Main Review

## DESCRIPTION

The `del_auction` docs define a stale auction as "one that has been running for 500 blocks without being filled". Partially filled auctions can be deleted too. After a partial fill, `scale_auction` keeps the original `block` on the remaining auction, and `delete_stale_auction` only checks `auction.block + 500 > sequence`. Once 500 blocks have passed since the auction started, anyone can delete the remainder, even right after a partial fill.

Keepers relying on the docs can expect a partially filled auction to stay open, while its remainder can be deleted at any time and the liquidation then has to be created again.

## RECOMMENDATION

Describe a stale auction as one that started at least 500 blocks ago, whether or not it has been partially filled.

## RESOLUTION

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

## INFORMATIONAL I-10

RESOLVED

## Stale comment in the pool factory

CATEGORY Documentation

LOCATION [https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool-factory/src/pool\\_factory.rs#L86-L87](https://github.com/Templar-Protocol/blend-contracts-v2/blob/7a09b0e3d2281ebbe5f619d88c722595e6dd2cfa/pool-factory/src/pool_factory.rs#L86-L87)

REVIEW Main Review

### DESCRIPTION

`deploy` has two copies of the comment "verify max positions is at least 2 and less than 64". The first, above the `max_positions` check, was updated to say pools have at most 30 reserves and 60 positions. The second, above the `min_collateral` check, still says:

```
// verify max positions is at least 2 and less than 64
// pools have a max of 50 reserves, so 100 is the max number of positions
if min_collateral < 0 {
```

It does not describe the check below it, and the limits are wrong: `MAX_RESERVES` is 30 and the factory allows at most 60 positions. This gives wrong limits to anyone reviewing or changing the deployment checks.

### RECOMMENDATION

Replace the second comment with one that describes the `min_collateral` check.

### RESOLUTION

Status: Resolved.

<https://github.com/Templar-Protocol/blend-contracts-v2/pull/13>

## § 05

# Disclaimer.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian's position is that each company and individual are responsible for their own due diligence and continuous security. Guardian's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis.

Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract's safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

## § 06

# About Guardian.

Guardian is a smart contract security firm. Every engagement is worked by two independent teams competing on the same scope, each with its own tooling and methods, so no single reviewer's blind spot survives.

**\$45B+**DIGITAL ASSETS UNDER  
REVIEW**200+**

ENGAGEMENTS COMPLETED

**500+**CRITICAL & HIGH FINDINGS  
REPORTED

Reach us at [audits@guardianaudits.com](mailto:audits@guardianaudits.com).