

# Independent Portfolio Oversight: Methodology and Control Contract

[Overview](#) · [Research methodology](#) · [Interview guide](#) · [Data contract](#)

## 1. Objective and scope

The oversight extension challenges the saved 2014–2025 WRDS/CRSP ETF implementation study without rerunning or importing the primary optimizer, simulator, or validation functions. It reconstructs economic state, reconciles risk and attribution, checks relational integrity in SQL, and verifies that deliberate defects trigger the relevant controls.

The question is **whether specified research records and reported analytics are consistent with their declared data, accounting, and policy contract**. A pass is a technical evidence-release result, not an investment recommendation, actual trading gate, institutional model validation, or certification of suitability.

The original research remains an exposure-implementation experiment: eight retrospectively selected ETFs, ten hypothetical account scenarios, three allocation methods, a frictionless monthly policy benchmark, and modeled execution costs. Source returns are actual current-vintage CRSP observations. Client restrictions, policy weights, initial wealth, and trading-cost assumptions are experimental inputs.

## 2. Independence boundary

`oversight_core.py` does not import production portfolio, simulation, or validation functions. The separate `oversight_contract.json` declares the expected universe, target and initial weights, methods, client bounds, dates, configuration, tolerances, and primary source-code hashes.

The checker independently:

- Builds the required security/date and account/date/method key sets.
- Propagates opening sleeve values with source total returns.
- Applies the saved trade dollar changes and recomputes costs from their notionals.
- Rebuilds account NAV, returns, benchmark, contributions, and interim drift.
- Reconstructs covariance windows and tracking errors.
- Evaluates saved records through separately versioned DuckDB SQL.

It does **not** independently obtain a second vendor feed, prove that the declared policy is appropriate, or certify the optimizer's global solution. The risk check uses the same established `sklearn.covariance.LedoitWolf` library, but independently selects the window and recomputes covariance and quadratic-form TE. For the risk budget, it reconciles the saved Minimum risk allocation's recomputed TE; it does not solve a separate optimization or prove minimum-risk

optimality.

The checker and primary path share source inputs, financial conventions, and an AI-assisted development process. This provides calculation-path independence, not organizational independence or elimination of common-mode errors. Human specification review remains necessary.

### 3. Inputs, grains, and provenance

| Input                      | Location                           | Required grain               |
|----------------------------|------------------------------------|------------------------------|
| Source returns             | data/wrds/daily.csv                | One date/ticker record       |
| Reference calendar         | data/wrds/calendar.csv             | One reference trading date   |
| Saved account performance  | outputs/wrds_private/daily.csv     | Date/client/method           |
| Saved allocation decisions | outputs/wrds_private/decisions.csv | Execution date/client/method |
| Saved trade ledger         | outputs/wrds_private/trades.csv    | Date/client/method/ticker    |
| Saved benchmark            | outputs/wrds_private/benchmark.csv | One evaluation date          |

The production configuration is compared to the declared contract. The empirical-run contract requires the saved run provenance and source manifest; cached source files are checked against that manifest. Artificial test fixtures explicitly opt out of empirical-source provenance. The checker records input file SHA-256 hashes; overrides have separate frame hashes. The runner additionally hashes protected input and research records before and after all incident experiments.

Original research files are not mutated by fault injection. Overrides are deep-copied in memory. A changed input must be investigated and versioned rather than silently treated as the original run.

Hashing establishes byte identity, not economic correctness or redistribution rights. The source contract in [DATA.md](#) governs return fields, historical identity, calendar, and licensing.

One explicit portability exception applies to **production source-code contract pins**: source text is normalized to LF line endings before hashing, so Windows CRLF and Unix LF checkouts can satisfy the same source contract. Input data, saved research files, and actual receipt file hashes remain raw-byte SHA-256 fingerprints. A normalized source pin and a raw-byte receipt hash serve different purposes and should not be compared as if they used the same preprocessing.

### 4. Structural controls and fail-closed behavior

Before pivoting or joining, the checker requires necessary columns, nonmissing keys, unique keys, finite numeric values, and valid timezone-naive normalized dates. It does not fill returns, drop duplicate rows, or silently shorten a period.

The source must contain the complete Cartesian product of declared tickers and reference trading dates. The evaluation panel must contain every date/client/method combination; decisions must exist for each declared monthly execution and account/method; trades must contain every expected ticker for those executions. A missing whole account or date is not accepted merely because remaining rows reconcile.

Additional controls include positive saved NAV, nonnegative saved costs, return values greater than  $-100\%$ , stable identity when identifiers are present, and supported CRSP flags when those fields are present. A total loss or delisting requires a separately implemented policy; neither is silently dropped or imputed.

Representative exact identifiers include `schema_*`, `nonnull_keys_*`, `unique_keys_*`, `valid_dates_*`, `finite_*`, `source_calendar_complete`, `complete_roster_*`, and `executed_configuration_drift`. The generated receipt is the authoritative list for a particular run.

Structural failure stops the affected reconstruction. Other economic discrepancies are accumulated so a reviewer can see related failures rather than only the first residual. An unexpected validator exception becomes `validator_execution = BLOCK`; that generic block is not sufficient evidence that an injected domain defect was correctly diagnosed.

## 5. Decision clock and independent risk checks

The execution dates are derived from the reference calendar as each evaluation month's first available trading close. The 756-observation baseline risk window ends strictly before execution, at the signal close. The execution day's return therefore cannot enter the decision's covariance estimate.

Checks reconcile signal date, history start, history count, shrinkage value, and trade/decision timestamps. `decision_chronology` checks strict ordering; `decision_risk_window` checks the actual expected window metadata. Both are needed: “before execution” alone does not prove the correct lookback window.

For proposed weights  $w$ , policy  $b$ , and annualized covariance  $S$ :

```
active_weights = w - b
estimated_TE = sqrt(active_weights' × S × active_weights)
S = daily_covariance × 252
```

The checker verifies covariance finiteness/positive semidefiniteness, full investment, bounds, saved TE, minimum-reference consistency, risk-budget definition, and the lower-turnover portfolio's budget. The baseline allowance is 25 bp of annualized TE above minimum feasible estimated TE—not 25% more risk, variance, or a limit on realized annual underperformance.

Risk-model validation is separate from future forecast accuracy. A correctly calculated ex-ante risk estimate can differ materially from subsequently realized risk.

The ex-ante reference is the fixed policy vector at the proposal. The realized tracking-error statistic instead uses earned account returns minus the independently reconstructed, monthly rebalanced and otherwise drifting benchmark returns. The evolving holdings, forecast uncertainty, legacy first-day transition, and repeated decisions mean these are different estimands. The allowance on one proposal is not a bound on the full-history realized TE difference between two implementation methods. The current checker reconciles each definition; it does not claim a formal risk-forecast calibration test.

## 6. Independent holdings and funded-cost reconstruction

The account state is total-return sleeve dollars. It is not a share, settlement, cash, or tax-lot ledger. Let  $h_{\text{prev}}$  be opening sleeve values,  $r$  source total returns,  $t$  trade saved signed trade notionals, and  $c$  the specified per-dollar trading-cost rate.

```
gain = h_prev * r
pretrade_holdings = h_prev + gain
fee = c * sum(abs(trade))
posttrade_holdings = pretrade_holdings + trade
prior_NAV = sum(h_prev)
closing_NAV = sum(posttrade_holdings)
gross_return = sum(gain) / prior_NAV
net_return = closing_NAV / prior_NAV - 1
```

On nonrebalance dates  $\text{trade} = 0$  and  $\text{fee} = 0$ . At an execution close the checker compares reconstructed pretrade and posttrade holdings to saved holdings, recomputed costs to saved costs, and closing weights to proposed weights. It also requires:

```
sum(trade) + fee = 0
closing_NAV + fee = pretrade_NAV
```

The primary simulator solves post-cost wealth before allocating. The checker deliberately does not call that funding solver; it challenges saved trades and their resulting accounting instead. This detects internally inconsistent source records while preserving a different calculation path.

The benchmark is independently propagated from target weights, reset frictionlessly at the same execution closes, and allowed to drift between those closes. The account begins at the declared legacy allocation. Old account holdings earn the first execution day's return before the initial transition; the resulting small unrestricted tracking difference is retained.

### Denominator policy

The exact return reduction is  $\text{fee} / \text{prior\_NAV}$ . A trading-cost-rate diagnostic uses  $\text{fee} / \text{pretrade\_NAV}$ ; gross traded fraction also uses pretrade NAV. These denominators answer different questions and are checked separately when the saved diagnostic fields exist.

Bounds apply at execution closes. Interim market-driven drift is independently measured and reconciled, not mislabeled as an execution-policy breach or silently corrected with extra trades.

## 7. Daily attribution and exact multiperiod linking

Daily asset contributions use actual opening sleeve values:

```
asset_contribution[i,t] = h_prev[i,t] * r[i,t] / prior_NAV[t]
cost_contribution[t] = -fee[t] / prior_NAV[t]

sum(asset_contribution[:,t]) = gross_return[t]
sum(asset_contribution[:,t]) + cost_contribution[t] = net_return[t]
```

Benchmark asset contributions use benchmark opening holdings and benchmark prior NAV. Subtracting those contributions from account contributions, then adding the negative account cost contribution, reconciles to daily net arithmetic active return.

Daily arithmetic contributions cannot simply be added to claim cumulative return. The extension uses **initial-capital-scaled dollar P&L**:

```
linked_portfolio_asset[i] = sum_t(h_prev[i,t] * r[i,t]) / initial_NAV
linked_benchmark_asset[i] = sum_t(b_prev[i,t] * r[i,t]) / initial_NAV
linked_cost = -sum_t(fee[t]) / initial_NAV

sum_i(linked_portfolio_asset[i]) + linked_cost
= terminal_portfolio_NAV / initial_NAV - 1

sum_i(linked_portfolio_asset[i] - linked_benchmark_asset[i]) + linked_cost
= (terminal_portfolio_NAV - terminal_benchmark_NAV) / initial_NAV
```

This is equivalent to scaling each day's contribution by that portfolio's prior wealth relative to initial capital. It telescopes because trades are self-financing and there are no external account cash flows. The benchmark uses its own evolving capital base, not the portfolio's.

Controls `daily_asset_gross_identity`, `daily_net_contribution_identity`, `daily_active_contribution_identity`, and `multiperiod_*_linking` test these relationships separately.

This is a declared accounting/P&L decomposition, not Brinson, factor, selection-alpha, or tax attribution. The linked fee term is not the complete impact relative to a separately simulated cost-free counterfactual: fees also change subsequent invested capital and gains.

## 8. Independent SQL controls

The versioned SQL files use DuckDB over saved records, not over Python reconstruction results:

- `sql/oversight_trade_controls.sql` aggregates trade rows to date/client/method, checks position arithmetic, fee formula, self-financing, saved daily NAV/cost agreement, and missing joins to daily and decision records.
- `sql/oversight_daily_controls.sql` uses `LAG(nav)` partitioned by client/method, with initial capital for each first row. It reconciles NAV-implied net return, the prior-NAV cost deduction, gross/net active returns, and benchmark alignment through a left join.

Structural key, finiteness, and coverage controls precede these queries. This matters because `SUM` can ignore `NULLs`, a one-to-many join can multiply values, and an inner join can hide unmatched dates. The SQL queries preserve unmatched records with left joins and report missing-match counts.

SQL identities can pass on a consistently altered ledger, while source-driven reconstruction can still fail. Conversely, relational checks can catch ingestion and reporting problems before an economic calculation is attempted. Neither replaces the other.

## 9. Tolerance policy

The baseline contract uses explicit absolute tolerances:

| Quantity                          | Tolerance                   | Interpretation                                           |
|-----------------------------------|-----------------------------|----------------------------------------------------------|
| Dollar reconciliation             | 1e-4 dollars                | \$0.0001 numerical residual, not a materiality allowance |
| Return/attribution reconciliation | 1e-10 decimal return        | Much smaller than an investable basis-point difference   |
| Weight feasibility/drift          | 2e-6 decimal weight         | Approximately 0.0002 percentage points                   |
| Risk-budget feasibility           | 2e-6 decimal annualized TE  | Approximately 0.02 bp of annualized TE                   |
| Risk-value reconciliation         | 1e-10 decimal annualized TE | Numerical agreement with the independent calculation     |

Each receipt records count, maximum absolute residual, tolerance, detail, and PASS or BLOCK. Contract validation rejects negative, nonfinite, or looser-than-approved tolerances. A numerical tolerance is not permission to accept a materially wrong result. These absolute tolerances are scoped to this \$1 million-initial-capital study; extension to very different account scales or numerical representations should review the tolerance policy rather than copying it without analysis.

## 10. Deliberate incident experiments

`run_oversight.py` first requires a clean baseline. It then runs five mutations separately on copies, requiring a domain-specific failed control in addition to an overall block:

| Case    | Deliberate mutation                                                      | Required detector family          | Correction/retest concept                |
|---------|--------------------------------------------------------------------------|-----------------------------------|------------------------------------------|
| INC-001 | Substitute ex-distribution returns for total returns                     | Daily return/NAV reconstruction   | Restore the declared total-return field  |
| INC-002 | Duplicate one trade-key row                                              | unique_keys_trades                | Restore a unique input snapshot          |
| INC-003 | Set one signal date equal to execution date                              | Chronology or exact risk window   | Restore the prior-close decision clock   |
| INC-004 | Move 1 percentage point from SPY to prohibited GLD in a No gold proposal | decision_constraint_bounds        | Restore the constrained proposal         |
| INC-005 | Add \$100 to one saved trade fee without changing notional or holdings   | Trade cost-formula reconciliation | Restore formula-consistent recorded fees |

INC-001 holds the saved trade ledger fixed while substituting a wrong return definition. The resulting disagreement is a diagnostic illustration, **not** a self-consistent alternative strategy backtest, observed loss, or measured investment impact. Any aggregate values from a blocked mutated run are diagnostic only and must not be ranked as valid portfolio results.

The runner retests the original clean inputs after all injections and confirms protected file hashes are unchanged. Because mutations occur in memory, “corrected retest” means rerunning the unchanged original specification, not repairing production files or claiming a real incident was remediated. Suggested operational owners in the receipt are illustrative responsibilities, not named people or active assignments.

A test case closes only if it was blocked by an expected domain control, the clean retest passes, and originals remain unchanged. An exception producing a generic block is not a successful domain-detection test. The five cases demonstrate detection of those specific defects; they do not

measure general detection sensitivity or cover every conceivable failure.

## 11. Historical stress readout

The runner also summarizes two already-exposed historical windows: the 2020 COVID selloff and the 2022 stock/bond drawdown. The windows are defined as returns after the stated starting close through the ending close. Each output states the start-exclusive/end-inclusive convention and session count.

Returns compound the saved net account path. Active cumulative return is the difference between cumulative account and benchmark returns. Window drawdown is measured relative to wealth within that selected window, including its starting wealth of one; it is not necessarily the full-history maximum drawdown.

These are descriptive historical scenario readouts, not newly untouched holdouts, a Monte Carlo distribution, forecast stress probabilities, or reverse stress tests. Their dates were chosen with knowledge of history.

## 12. API and artifacts

The Python interface is:

```
run_validation(root: Path, overrides: dict | None = None,
               output_private: Path | None = None) -> dict
```

Overrides may replace the declared input tables or contract for tests. The result is JSON-safe and contains status, checks, blocker IDs, aggregate metrics, provenance, and limitations. Detailed reconstructed rows are omitted from public JSON. If explicitly requested with `output_private`, the resolved destination must be under the repository's `outputs/oversight_private/` directory; any other destination blocks validation. Economic failures can retain detail there for diagnosis, while structural failures stop before reconstruction.

Run the saved-data oversight workflow from the repository root:

```
python run_oversight.py
```

This requires the existing licensed source and detailed local research outputs. It makes no fresh market-data request and has no synthetic fallback for empirical results. Hand-derived test fixtures remain artificial and are labeled as such.

Public aggregate evidence is stored under `outputs/oversight/`:

- `validation_summary.json`: clean-run checks, aggregates, and input provenance.
- `incident_summary.json`: injection definitions, domain detections, diagnostic aggregate impact, and clean retest.
- `historical_stress.json`: labeled, already-exposed historical windows.
- `run_receipt.json`: timing, input/code hashes, unchanged-input result, and technical release status.
- Workbook, memo, dashboard, and interview-guide renderings: presentation artifacts built from aggregate evidence and clearly labeled synthetic illustrations.

Detailed reconstructed holdings and contributions, raw CRSP rows, and detailed primary paths remain under ignored local directories. A public workbook's illustrative trade calculation is synthetic; it is not a republished security-day source record. A reader can review code and aggregate evidence publicly, but a full empirical rerun requires their own data entitlement and local inputs.

## 13. Interpretation and limitations

A passing receipt means the stated controls passed for identified artifacts. It does not establish future alpha, model suitability, complete data correctness, production readiness, tax savings, or independent optimizer optimality.

The research uses a current-vintage extract and a fixed retrospectively selected universe. Lagged chronology does not eliminate database revisions, universe-selection concerns, or the fact that the history has already been examined. ETF total-return sleeves omit live shares, tax lots, external cash flows, real order handling, market impact, settlement, and household restrictions. The cost rate is modeled rather than observed.

The useful result is a transparent, reproducible separation of **investment trade-offs**, **accounting/control evidence**, and **operational exceptions**. Keeping those categories distinct is the core oversight discipline.

For worked calculations and source-level exercises covering these distinctions, see the [question-first interview rehearsal](#). Its invented two-asset figures are teaching fixtures, not additional empirical portfolio results.