Waste Reduction Playbook

The Waste Reduction Playbook is a practitioner's guide for cutting the compute spend that produces no quality value — redundant retries, duplicate retrieval, runaway loops, idle infrastructure, and over-provisioned tier selections. It does not optimize quality-critical steps (those are governed by e6-quality-cost-matrix and its demotion-waiver process); it targets the structural waste that accumulates invisibly in production agent stacks. The playbook is organized as a set of detection patterns, each paired with a concrete remediation procedure and a cap or limit to enforce going forward.

Waste Pattern Catalogue

W1 — Redundant retries

What it is. The agent retries a failed tool call or model call beyond the point where retrying is productive, accumulating cost without progress. Distinct from looping-retry (failure mode in s1-failure-mode-register), which is a planner-level loop; redundant retries are connector-level retries that keep firing after the root cause is known.

Detection signal. tool_calls[] in s2-audit-trail-schema shows the same tool_id called more than max_retries times within a single run window, or a cumulative est_cost_usd for retries exceeds 20% of the run's total cost.

Remediation.

1. Set max_retries in the e6-routing-policy fillable block to the value validated by the s1-fallback-architecture-blueprint retry caps for this connector class (typically 2–3 attempts).

2. Enforce an exponential backoff starting at backoff_seconds before the first retry.

3. After max_retries, route to the declared fallback rather than continuing to retry. Log the fallback via C7 with fallback_tier_if_any populated.

4. If the connector fails max_retries attempts, emit an escalation event to s1-hitl-review-policy — not a silent failure.

Cap to enforce. max_retries ≤ 3 for any single tool call within a run. Any override requires a written entry in the e6-routing-policy configuration with rationale.

W2 — Duplicate retrieval

What it is. The same or semantically equivalent retrieval query is executed more than once within a run or across closely-spaced runs, paying embedding and search costs repeatedly for identical results.

Detection signal. Two or more entries in s2-audit-trail-schema tool_calls[] with the same tool_id (a retrieval connector) and a query embedding cosine similarity > 0.97 within a single run; or the same query appearing in runs within the cache window defined in e6-routing-policy.

Remediation.

1. Enable the cache block in e6-routing-policy with similarity_threshold: 0.97 (or tighter for high-stakes steps).

2. Set max_cache_age_seconds to match the freshness requirement of the source's confidence_band from e6-cost-model. A high confidence-band source with nightly refresh tolerates an 8-hour cache window; a real-time telemetry source should be in excluded_step_ids.

3. For intra-run duplicates, add a run-scoped deduplication buffer in the orchestration layer: before any retrieval call, check whether an equivalent query was executed earlier in this run's tool_calls[] history.

Cap to enforce. No retrieval connector call may be duplicated within the same run unless the step_id is in excluded_step_ids (justified by freshness requirements). Violations surface in the C7 audit records as cost waste without quality gain.

W3 — Runaway loops

What it is. A planner-executor cycle repeats without meaningful state change — the planner generates a new plan step, the executor runs it, the result is identical to the prior step's result, and the planner generates the same step again. Distinct from looping-retry (which tracks the same call repeated); runaway loops track semantic repetition across the planning cycle.

Detection signal. decision_trace[] in s2-audit-trail-schema contains the same step value (or a step with identical outputs) three or more consecutive times. s1-failure-mode-register failure mode looping-retry captures the connector-level version; the planner-level version is detected here.

Remediation.

1. Implement a state-hash check in the orchestration layer: before executing any planner-generated step, compute a hash of the current state and compare to the prior state. If identical, abort the planning cycle and route to fallback.

2. Set a hard loop cap at 10 total planning cycles per run (override requires configuration entry). This is the routing-policy-level enforcement of the s1-fallback-architecture-blueprint loop-break pattern.

3. On loop-break, emit a C7 record for the truncated call with routing_rationale: "loop-break: step-hash repeated N times" and escalate via s1-hitl-review-policy.

Cap to enforce. Maximum 10 planning cycles per run. Hard cap enforced by the routing policy; soft warning at 6 cycles.

W4 — Over-tiered calls

What it is. Steps are routed to a higher model tier than their quality requirement demands, either because the quality-cost matrix was not populated, because a developer defaulted all calls to premium, or because a one-off integration bypassed the routing policy.

Detection signal. C7 records in s2-audit-trail-schema show model_tier: premium for a step whose minimum_tier in e6-quality-cost-matrix is standard or lower. Measured as the percentage of premium-tier calls that could have been served at a lower tier without quality degradation. A rate above 10% of premium calls being over-tiered is a waste indicator.

Remediation.

1. Audit the C7 records monthly. For each premium-tier call, check the step_id against e6-quality-cost-matrix. Any mismatch is a misconfiguration.

2. Trace the bypass: did the call go through the routing policy? If not, enforce that all model calls must pass through e6-routing-policy — no direct provider API calls from agent code.

3. Update the quality-cost matrix if the step was newly added without a matrix entry. Un-registered steps must default to premium (safe default) but trigger a "unregistered step" warning in the C7 record.

Cap to enforce. Zero direct model API calls outside the routing policy. Any integration detected bypassing the policy is a MAJOR configuration violation.

W5 — Unused or idle infrastructure

What it is. Vector stores, dedicated compute instances, or reserved model capacity provisioned for peak load but underutilized at baseline. Common after a burst deployment that is not right-sized post-launch.

Detection signal. Observability layer (from e6-cost-model observability block) shows vector store query rate below 20% of provisioned capacity for more than 14 consecutive days; or compute utilization below 15% of provisioned instances for more than 7 consecutive days.

Remediation.

1. Compare provisioned capacity to p95 observed throughput from the C7 records. Right-size to 150% of p95 (not peak) as the provisioned ceiling.

2. For vector stores, evaluate whether a serverless/on-demand tier fits the query-rate pattern. Reserved capacity is cost-efficient only above a utilization threshold (typically 60%+).

3. For compute, implement auto-scaling down to zero for batch task classes. Interactive task classes may retain a warm instance but should scale to zero outside operating hours.

4. Open a cost-model review ticket whenever idle-infrastructure waste exceeds 15% of the monthly compute line.

W6 — Expensive human-review routing

What it is. HITL gates fire too frequently because the confidence threshold is set too conservatively, because low-quality inputs are not filtered before reaching human review, or because the agent escalates rather than applying a declared fallback.

Detection signal. Human-review cost in e6-cost-model exceeds 30% of total monthly spend. Escalation rate from s1-hitl-review-policy is rising (escalation drift per s2-scoring-system drift indicators). HITL events in provenance records show a majority of rationale entries citing low-confidence rather than policy-mandated gates.

Remediation. This remediation must be handled carefully: reducing HITL frequency is only safe if the underlying quality has genuinely improved, not if the threshold was simply relaxed.

1. Analyze the HITL event log from s1-hitl-review-policy. Separate mandatory HITL gates (irreversible-impact boundary crossings) from confidence-triggered escalations.

2. For confidence-triggered escalations, evaluate whether improving retrieval quality (raising confidence_band of sources) or adding a pre-filter deterministic step would reduce escalation rate while maintaining quality.

3. Do not lower confidence thresholds without a corresponding eval showing quality has improved. Threshold changes are governed by s1-threshold-escalation-spec.

4. Route low-quality inputs (e.g., malformed applications, obviously out-of-scope queries) to a deterministic rejection before they reach the model or the human queue.

Remediation Prioritization

When multiple waste patterns are active, address them in this order:

Priority	Pattern	Why first
1	W3 Runaway loops	Unbounded spend risk; a loop can consume the entire budget ceiling
2	W1 Redundant retries	Connector-level waste that compounds quickly on flaky external APIs
3	W4 Over-tiered calls	Often the highest per-unit waste; easy to detect from C7 records
4	W2 Duplicate retrieval	Significant at scale; cache configuration resolves most cases
5	W6 Expensive HITL	Requires quality evidence before reducing; higher caution
6	W5 Idle infrastructure	Lowest urgency; right-sizing is a planning exercise, not an emergency

Worked Example: GoodCombinator Agent Audit (illustrative)

A quarterly audit of the GoodCombinator.ai partner-matching agent surfaces the following waste profile. All figures are (illustrative).

Waste pattern	Detection	Estimated monthly waste	Remediation
W1 — Redundant retries	LinkedIn connector retried 6× per failed call (max_retries=6, should be 3)	$28/mo	Set max_retries=3; add exponential backoff

Waste pattern	Detection	Estimated monthly waste	Remediation
W2 — Duplicate retrieval	34% of embedding queries duplicated within-run; cache disabled	$45/mo	Enable cache at 0.97 similarity threshold; exclude partner-profile steps
W3 — Runaway loops	Two planning cycles hit 8+ iterations/run	$19/mo	Add state-hash check; set hard cap at 10 cycles
W4 — Over-tiered	22% of partner-brief-draft calls going to premium despite standard minimum_tier	$61/mo	Enforce routing-policy path for all calls; audit direct API call in legacy connector
W5 — Idle infra	Vector store at 12% utilization; provisioned for 3× actual query rate	$38/mo	Downgrade to on-demand vector tier
W6 — HITL overuse	41% of escalations are confidence-triggered, not policy-mandated	$94/mo	Improve retrieval source quality; defer threshold change pending eval

Total recoverable waste: ~$285/mo (illustrative), approximately 28% of the $1,010/mo total spend. The playbook does not target the remaining 72%, which is quality-producing spend confirmed by the quality-cost matrix.

Maintenance Cadence

Run a waste audit against C7 records at least monthly. Use the s2-regression-discipline cadence as a synchronization point: if an eval regression surfaces alongside a cost spike, investigate over-tiering or loop escalation first. Any waste remediation that changes the max_retries, cache configuration, or loop cap in e6-routing-policy requires a version bump in the routing policy and a note in e6-compute-economics-package.