SALESFUSION

Executive Control for Business Agents

Technical architecture and implementation report

PUBLIC RELEASE · VERSION 1.0.0 · 2026-07-17

Executive Control for Business Agents

Publication metadata

Field	Value
Author	SalesFusion Research
Publication date	2026-07-17
Version	1.0.0
Canonical slug	executive-control-for-business-agents
Canonical path	/technical-papers/executive-control-for-business-agents
Publication class	Technical architecture and implementation report
Approval state	Founder-approved public release

Revenue Cortex connection

Revenue Cortex uses executive control to decide what revenue work should run, pause, escalate, or remain blocked. In a growth system, this includes selecting the next buyer segment, approving outbound claims, routing sales opportunities, holding unsafe automation, and preserving human authority over consequential customer-facing actions.

Evidence-status box

> This paper proposes an executive-control layer for business agents and describes selected SalesFusion controls that have been implemented as operating patterns: authority boundaries, approval separation, decision records, and first-job gates. The complete executive-control architecture requires expert-scored scenario evaluation before claims about decision quality or operational effectiveness are appropriate. The allowed claim is architectural: business agents need an executive-control layer that prioritizes, inhibits, escalates, and allocates work under explicit authority and constraints.

Abstract

Business agents are often evaluated by whether they can complete a task. In operating environments, task completion is not enough. The harder question is whether the agent should attempt the task, under which objective, with which evidence, within which authority, after asking which question, and with what rollback path. We define executive control as a layer between Company Memory and workflow execution. It builds objective

trees, detects constraints, ranks options, selects next best actions or questions, inhibits unsafe or low-evidence work, escalates authority gaps, and produces decision packets. The layer treats do_nothing, collect_evidence, block, and escalate as valid outputs, not failures. This paper specifies data objects, priority rules, inhibition logic, authority checks, interfaces, evaluation scenarios, and failure modes for executive control in governed business AI systems.

The problem

An agent with tools can act faster than a team can review every intermediate step. That speed is useful only when the agent can decide when not to act. Business settings contain conflicting goals, stale context, missing evidence, irreversible actions, external consequences, approvals that expire, and tasks that create activity without advancing an objective. A workflow runner cannot resolve those issues alone. A language model prompt can list rules, but prompts are not durable governance.

The common failure is direct task execution. A request enters the system and the agent tries to satisfy it. The agent may ignore the current objective, choose a visible but low-value task, ask the wrong human for approval, reuse stale context, or continue a run after the correct answer is to stop. When recommendations and execution are collapsed, the system has no clean place to inspect tradeoffs or enforce authority.

Executive control is the missing layer. It is not a manager persona. It is a stateful control system that converts memory, objectives, constraints, and policies into a decision packet before execution.

Contributions

This paper contributes a technical model for executive control in business agents.

First, it defines objective trees, constraints, anti-goals, priority factors, authority classifications, and decision packets.

Second, it specifies next-best-action and next-best-question selection as separate outcomes.

Third, it defines inhibition and escalation rules that block action under missing evidence, unresolved conflict, stale context, duplicate work, insufficient authority, or unsafe reversibility.

Fourth, it describes implementation interfaces connecting memory snapshots, policy evaluation, decisions, approvals, workflows, and telemetry.

Fifth, it proposes an evaluation suite for constrained business decisions without claiming field results.

Definitions

Objective tree is a hierarchy of desired outcomes, subgoals, metrics, constraints, anti-goals, and tradeoff rules.

Constraint is a binding condition that limits action. Constraints can be policy, capacity, legal, financial, security, data-quality, timing, or objective-conflict constraints.

Priority is the controller's ordering of candidate work under objective value, urgency, evidence quality, reversibility, dependency, cost, risk, and authority.

Next best action is the highest-ranked allowed action after constraints, evidence, and authority checks.

Next best question is the highest-value question when action is blocked by missing, conflicting, stale, or restricted information.

Inhibition is an explicit decision to prevent or delay an action.

Escalation is routing a decision to a human or role with the required authority and context.

Decision packet is the inspectable artifact that contains objective, state snapshot, evidence, options, constraints, recommendation, authority result, expected artifacts, reversibility, and confidence.

Executive-control model

The controller receives a request or trigger and does not immediately execute it. It performs six operations:

1. Build a state snapshot for the relevant objective, effective time, freshness requirement, and actor.
2. Resolve the objective tree and identify constraints, anti-goals, dependencies, and metric definitions.
3. Generate options, including non-action options such as do_nothing, collect_evidence, block, and escalate.
4. Score options using transparent factors and reject options that violate hard constraints.
5. Evaluate authority and approval requirements for the selected option.
6. Produce a decision packet or route an approval request.

A minimal objective tree:

```
{
  "objective_id": "obj_operating_motion_quality",
  "outcome": "improve an approved operating motion",
  "metrics": ["metric_quality", "metric_cycle_time"],
  "constraints": ["no_external_mutation_without_approval"],
  "anti_goals": ["activity_without_outcome_signal", "policy_bypass"],
  "priority_rules": [
    "hard_constraints_override_score",
    "fresh_source_required_for_consequential_action",
    "irreversible_action_requires_escalation"
  ]
}
```

A candidate option can be represented as:

```
{
  "option_id": "opt_collect_evidence",
  "type": "collect_evidence",
  "objective_refs": ["obj_operating_motion_quality"],
  "required_assertions": ["asrt_current_constraint"],
  "expected_artifact": "updated_state_snapshot",
  "consequence_class": "read_only",
  "reversibility": "not_applicable",
  "authority_required": "analyst",
  "blocked_by": []
}
```

Priority is not a single opaque score. The controller should preserve factor values:

Factor	Question
objective value	Which objective or constraint does this advance?
urgency	Does delay change the state or opportunity?
evidence quality	Are required assertions active, fresh, and sourced?
reversibility	Can the action be undone and proven undone?
authority	Is policy clear and approval current?
dependency	Does another task need this result?
cost and capacity	Is the work bounded by budget and operator capacity?
risk	What is the downside of wrong execution?

Hard constraints override score. If a highly valuable action lacks authority, the answer is escalation, not execution.

Inhibition, escalation, and decision rules

Inhibition rules are explicit and testable:

Condition	Controller output
required source is stale	collect evidence
required assertion has unresolved conflict	block or escalate
action is outside actor authority	escalate or deny
approval expired or version-mismatched	require new approval
action is duplicate of a completed receipt	return prior receipt
action is irreversible and rollback absent	escalate before execution
objective conflict is unresolved	decision needed
no action is warranted	do_nothing with reason

The control loop:

```
snapshot = memory.snapshot(objective, actor, effective_time)
constraints = resolve_constraints(objective_tree, snapshot, policies)
options = generate_options(request, objective_tree, snapshot)

for option in options:
    option.blockers = detect_blockers(option, constraints, snapshot)
    option.authority = policy.evaluate(option.action_digest, actor, scope)
    option.score = score_priority(option, objective_tree)

eligible = options.without_hard_blockers()
if eligible.is_empty():
    return packet(output=best_inhibition(options))

selected = rank(eligible).first()
if selected.authority.requires_approval:
    return packet(output="escalate", approval_request=selected.authority)
return packet(output="ready", selected=selected)
```

The controller must explain why a rejected option was rejected. Without rejection reasons, a reviewer cannot distinguish good inhibition from model hesitation.

Authority and decision packets

Authority is evaluated through policy, not inferred from user tone. The system classifies actions as read-only, internal reversible, external reversible, external consequential, financial, legal, or security-sensitive. Unknown classification defaults to the stricter path and escalates.

Approval is bound to exact action, version, scope, approver, and expiry. It does not execute the action. Recommendation and decision are separate operations and may require separate principals. This pattern is consistent with risk framing from NIST AI RMF and resource-scoped access reasoning from NIST SP 800-207, but the references do not validate any SalesFusion-specific outcome.

A decision packet should include:

Field	Required content
objective context	objective refs, metrics, anti-goals
memory state	snapshot id, evidence, conflicts, unknowns, exclusions
options	considered actions and questions
recommendation	selected next move and rationale
inhibition log	rejected options and blockers
authority	policy result, approval requirement, consequence class
execution contract	expected artifact, receipt, idempotency key requirement
reversibility	rollback path or escalation reason
confidence	calibrated statement and limits

Decision packets are the main artifact of executive control. They let a human inspect the reasoning without reading every retrieved source or the full model transcript.

Implementation

The executive-control layer can be implemented as a service that sits between state snapshots and workflow runs. It should use stable APIs:

```
POST /v1/state/snapshot
POST /v1/policies/evaluate
POST /v1/decisions
POST /v1/decisions/{decision_id}/evidence
POST /v1/decisions/{decision_id}/recommend
POST /v1/approvals
POST /v1/workflows/{workflow_id}/simulate
POST /v1/workflows/{workflow_id}/runs
```

SalesFusion has implemented selected controls in its operating patterns: approval boundaries, first-job gates, run evidence, and decision-oriented delivery records. The full controller should add objective-tree resolution, transparent priority scoring, systematic inhibition, and expert-reviewable packets.

Implementation invariants:

Invariant	Reason
no source-free consequential recommendation	blocks generated fact from driving action by rule
no execution from approval event alone	blocks approval/execution collapse by rule
no policy bypass from prompt or source text	keeps instructions out of evidence
no hidden overwrite of objective or constraints	preserves reviewability
no completion without receipt contract	checks for the evidence required to reject false-completion claims

The controller should also log all decision events with trace IDs. OpenTelemetry concepts can support infrastructure observability, while domain events record decision evidence, approvals, blockers, and outcomes.

Evaluation

Executive-control evaluation should use scenario tasks scored by domain experts. Scenarios should include several valid tasks but one binding constraint, objective conflict, missing evidence, irreversible request, expired approval, low-value work with high activity, urgent request outside authority, and a case where no action is warranted.

Scoring dimensions:

Dimension	What reviewers inspect
objective alignment	recommendation tied to the right objective
constraint identification	hard constraints recognized and enforced
priority quality	option ranking is defensible
option completeness	question, block, and do-nothing options considered
inhibition correctness	unsafe or unsupported actions stopped
escalation correctness	right authority path selected
reversibility awareness	rollback or irreversibility handled
explanation traceability	packet links to sources and policy
calibration	confidence reflects evidence quality

Adversarial tests should include prompt requests to ignore policy, source documents containing instructions, approval references from another version, a principal trying to expand its own scope, and one tenant referencing another tenant's object. Pre-installation target for unauthorized action and cross-tenant leakage should be zero.

This evaluation can support claims about controller behavior on the test suite. It cannot support claims about revenue impact or general decision superiority without a separate field study.

Safety, risks, and failure modes

Executive control can fail by over-acting, under-acting, escalating to the wrong role, ranking proxy activity above outcome-relevant work, treating an approval as execution, accepting stale evidence, suppressing uncertainty, or producing packets that are too vague to review. It can also become a single point of policy error if the authority registry is wrong.

Controls include explicit consequence classes, fail-closed unknown authority, approval expiry, version binding, inhibition logs, separate recommendation and execution events, duplicate-action detection, and review queues. OWASP LLM application threats are relevant to prompt injection and tool misuse; the specific controls must be tested in context.

Human error remains. The controller can make a recommendation inspectable, but a human can still approve a poor action. The system should therefore preserve rejected, blocked, and deferred decisions for later review rather than only celebrating completed work.

Limitations

This paper does not present a validated benchmark result. It defines the controller and the evaluation needed. Priority scoring remains partly judgment-based until enough labeled decisions exist. Objective trees can be incomplete or internally inconsistent. Authority policy can lag real organizational responsibility. Some actions have ambiguous reversibility, and some outcomes are delayed enough that the controller cannot know whether a decision was good within the run window.

The architecture also assumes the memory layer can provide source-backed state. If Company Memory is weak, executive control will mostly produce better-labeled uncertainty rather than better actions.

Practical implications

Builders should place executive control before workflow execution. The first version does not need a complex optimizer. It needs a state snapshot, objective tree, consequence classifier, policy evaluation, option generator, inhibition rules, and decision packet.

Operators should expect the controller to ask questions and block work. A system that always returns an action is not exercising executive control. A good first deployment should include read-only scenarios, simulated runs, approval tests, and explicit no-action cases.

References used

This paper uses NIST AI Risk Management Framework for risk framing, NIST SP 800-207 for resource-scoped access reasoning, OpenTelemetry Specification for observability concepts, and OWASP Top 10 for Large

Language Model Applications for threat awareness. These references do not validate SalesFusion-specific executive-control effectiveness.

Conclusion

Business agents need more than memory and tools. They need a control layer that decides what not to do, what to ask, when to escalate, and how to bind a recommendation to authority and evidence. SalesFusion has implemented selected governance controls that support this direction, while the complete executive-control layer remains a proposed architecture requiring expert-scored evaluation. The practical standard is clear: before a workflow runs, the system should produce a decision packet that a competent reviewer can inspect, challenge, approve, reject, or send back for evidence.

Changelog

- 2026-07-17 — Version 1.0.0: founder-approved first public release; publication metadata, evidence-status box, limitations, and verified references retained.