Permission Architecture Spec

This spec defines the structural permission model for agents with tool authority: the principal taxonomy, role definitions, least-privilege assignment rules, tool-scope boundaries, and sensitive-data authorization ceilings. It translates the default-deny posture declared in t8-zero-trust-assumptions into a concrete, implementable structure that t8-tool-use-control-list populates with per-tool decisions. Every agent deployment instantiates this architecture with agent-specific values before any tool is allowed to execute.

1. Principal taxonomy

A principal is any identity that can request an action from the agent. The taxonomy has four levels, each carrying a different default authorization ceiling.

Agent-self: The agent acting autonomously on a declared task. This is the typical principal in a production run. Its authorization ceiling is the intersection of the operating context canvas risk_class floor table and the per-tool control list. The agent-self principal never self-elevates; it cannot modify the control list at runtime.

Human-operator: A named human reviewer or operator interacting through an approved interface (HITL review queue, admin panel, approved CLI). Operators may approve, modify, or refuse staged actions per s1-hitl-review-policy. They may not override the default-deny rule or the irreversible-impact boundary without a canvas version bump.

Upstream-agent: An orchestrating agent or workflow runner that invokes this agent as a sub-agent or tool. Upstream-agent principals are treated with the same skepticism as untrusted callers until they present a valid session token and their identity resolves in the principal registry. An upstream-agent receives at most the same authorization ceiling as agent-self for the same tool; no upstream agent may grant itself higher authority than the originating canvas declares.

Service-account: A non-interactive service or connector that pushes data or triggers runs (webhooks, scheduled ingestions, event-driven invocations). Service accounts may only trigger read and external-write actions on explicitly listed connectors; they may not initiate financial, comms, admin, or destructive actions.

2. Least-privilege assignment

Least privilege means every principal holds the minimum permissions required for its declared task scope and nothing more. This produces a four-step assignment discipline.

Step 1 — Declare the task scope. From the agent_objective field on the operating context canvas, enumerate every tool invocation required to complete the task in the normal (non-degraded) path. This is the maximum permission set the agent should ever need.

Step 2 — Strip non-task tools. Any tool reachable by the agent runtime that is not required by the declared task scope is listed as blocked in the control list. This includes utility tools, debugging connectors, and any tool inherited from a base configuration.

Step 3 — Assign control level by action class. For each required tool, assign the minimum control level from the risk_class floor table in t8-zero-trust-assumptions that satisfies the task. Do not assign auto when confirm is sufficient. Do not assign confirm when human-approval is required by the action class floor.

Step 4 — Time-bound and scope-bound. Permissions apply to the current declared task only. A calendar-write permission granted to complete a guest check-in does not authorize a separate calendar delete in the same run session. Scope is task-granular, not session-granular.

3. Role definitions

Roles bundle permissions for common agent archetypes. A concrete agent instantiates one or more roles plus any task-specific overrides declared on the canvas.

Role	Permitted action classes	Notes
read-only-analyst	read	No writes of any kind. Safe for RAG, research, triage classification.
draft-producer	read, external-write (draft)	May write to a draft/staging area; may not publish or send.
comms-operator	read, comms (with HITL gate)	Guest messages, applicant notifications. Every send requires an open HITL gate.
financial-reader	read (financial data)	Read balance, payout, invoice data. Never write financial records.
workflow-executor	read, external-write (transactional)	May commit records to a system of record when HITL gate is open.
admin-agent	read, admin (with HITL gate)	User provisioning, connector credential rotation. Blocked without open HITL.

No role includes destructive. Destructive actions require explicit per-tool listing in the control list at blocked or human-approval only, and only after a canvas owner sign-off.

4. Tool-scope boundaries

A tool-scope boundary defines the narrowest possible authorization for a given connector. It is expressed as a tuple: (connector, operation, resource_scope, data_class_ceiling).

- connector: the named integration (e.g., ownerrez-api, gmail-oauth, stripe-payout, swc-stormwater-portal)
- operation: the specific HTTP method or SDK call permitted (GET /reservations/{id}, not PATCH /reservations/)
- resource_scope: the resource ids or patterns in scope (e.g., property ID PP-001, not all properties on the account)
- data_class_ceiling: the highest sensitivity class of data this tool may touch (public, internal, restricted, pii)

A tool-scope boundary for a broader operation than the task requires is a defect, not a feature. The control list enforces the narrowest declared boundary; the agent runtime must pass the exact resource_scope to the connector, not a wildcard.

5. Sensitive-data authorization ceilings

Four data classes are recognized, ordered by sensitivity. A principal may be authorized for a class and all classes below it; authorization for a higher class is never inferred from a lower authorization.

Data class	Examples	Authorization requirement
public	Published permit status, public parcel data, public STR listing info	agent-self, no gate required
internal	Booking confirmations, rate schedules, operational logs, draft documents	agent-self + read scope
restricted	Financial records, owner/operator PII, compliance determinations, audit trails	human-operator approval per read; never in model context without redaction
pii / sealed	Guest PII, applicant identity data (FS Chapter 119 exemptions [VERIFY]), healthcare	human-operator approval + authorized sink only; pii-leak-risk gates egress

When an action would route restricted or pii data to a connector, the permission-decision step checks that the connector is in the authorized-sinks list for that data class. If not, the action is blocked and a pii-leak-risk failure mode fires per s1-failure-mode-register.

6. Permission inheritance and override rules

Permission grants do not accumulate across runs, agents, or callers:

- A human operator approval in run N does not pre-authorize the same action in run N+1. Authorization is per-run.
- An upstream-agent cannot grant permissions it does not itself hold. The effective permission is the minimum of the upstream-agent's ceiling and the target tool's control-list row.
- A service-account trigger cannot elevate permissions for the triggered run beyond the service account's own ceiling.

Permission downgrades are always safe and may be applied without a canvas version bump. Permission upgrades (moving from blocked to any non-blocked state, or from confirm to auto) require: (a) a task-scope justification, (b) a canvas version bump if the floor table is affected, and (c) a note in the control list documenting the rationale.

7. Worked example: Good Combinator AI-for-Good Platform

The Good Combinator platform operates an agent that manages grant-pipeline workflows for civic AI projects. The agent has tool authority over: Notion (read and draft), Gmail (comms), an Airtable CRM (read/write), a Stripe donor-management connector (read-only), and Dropbox (read internal documents). The platform is risk_class: high.

Applying least-privilege assignment:

- Notion: task scope is read existing pages and write draft pages in the /Grants workspace only. Tool-scope boundary: (notion-api, GET + POST /pages, workspace: grants-workspace-id, internal). Control level: confirm for writes.
- Gmail: task scope is draft outbound grant-status notifications for human review before send. Tool-scope boundary: (gmail-oauth, POST /drafts, sender: grants@goodcombinator.ai, internal). Control level: human-approval (comms floor at high).
- Airtable CRM: task scope is read grant pipeline records and update status field only. Tool-scope boundary: (airtable-api, GET + PATCH /records/{id}.status, base: grant-pipeline-base, internal). Control level: confirm.
- Stripe: task scope is read donor aggregate totals only — no write, no per-donor PII. Tool-scope boundary: (stripe-api, GET /balance + GET /payouts, account: goodcombinator-platform, restricted). Control level: confirm; a restricted ceiling means the data never enters the model context in raw form.
- Dropbox: task scope is read internal grant templates. Tool-scope boundary: (dropbox-api, GET /files/{id}, folder: /Grant Templates, internal). Control level: auto.

Any tool not in this list — including a hypothetical Slack connector or a payments write call — is blocked by default-deny. The agent runtime enforces this before any model invocation attempts the call.

8. Maintenance cycle

Permission architecture is reviewed at three triggers: (a) any new tool or connector added to the agent runtime, (b) any change to the operating context canvas risk_class, and (c) the quarterly adversarial-suite review cadence in s1-monitoring-rollout-postmortem. Permissions granted 60+ days without any invocation are candidates for removal under the stale-permission rule in t8-zero-trust-assumptions. Each review is documented as a canvas version bump if any floor-table entry changes, or as a control-list minor update if only individual tool rows change.