

ENTERPRISE AI GOVERNANCE

Administration Agents

Governing the enterprise AI platform with
autonomous administrative agents

AUTHOR

Neurologic AI

PUBLISHED

September 2026

REFERENCE PLATFORM

Corporate GPT

ABSTRACT

As enterprises move from single chat assistants to fleets of purpose-built AI agents, the hardest problem is no longer answering questions; it is governing the system that answers them. Who may access which knowledge? Which connectors are live? Who built which agent, and who may run it? Where did a piece of personal data come from, and can it be erased on request? This white paper introduces **Administration Agents**: the class of autonomous and semi-autonomous agents, together with the governance surface they operate on, that administer an enterprise AI platform. Using Corporate GPT, an on-premises retrieval-augmented generation (RAG) and agent-building platform, as the

reference implementation, we describe how administrative work (identity and access, knowledge onboarding, agent curation, usage and cost oversight, and privacy and audit compliance) can be delegated to agents that operate under strict, fail-closed guardrails. We give particular attention to a generative form of administration: agents that mine a user's own recurring work patterns and proactively propose a purpose-built agent or workflow to automate them. We conclude with the design principles that make such delegation safe.

KEYWORDS

Enterprise AI governance

Administration agents

RBAC

Access control lists

Agent builder

Auditability

PII compliance

Retrieval-augmented generation

01

Introduction

The first wave of enterprise generative AI delivered a single assistant behind a chat box. The second wave, already underway, delivers a platform: many agents, drawing on many knowledge sources, invoked by many people across an organisation, each with different clearances and different needs. Corporate GPT is one such platform. It retrieves answers from a company's own documents, databases, ticketing systems and collaboration tools; it lets non-developers compose their own agents from a curated palette of capabilities; and it runs entirely inside the customer's security boundary.

This shift multiplies the value of the platform, but it also multiplies the administrative surface. Every knowledge source has to be onboarded, scoped, and kept in sync. Every user has to be provisioned, placed in the right groups, and given exactly the access they need, no more. Every agent that a power user builds has to be validated, published, and shared under control. Every answer that touches personal or regulated data has to be traceable, and, when the law demands it, erasable. Done manually, this work does not scale: it becomes the bottleneck that keeps enterprise AI stuck in pilot.

The same agentic paradigm that powers the platform's answers can, and should, power **its administration**.

The thesis of this paper is that the same agentic paradigm that powers the platform's answers can, and should, power its administration. An **Administration Agent** is an agent whose domain is the platform itself: its users, its knowledge, its connectors, its agents, and its compliance obligations. Administration Agents observe platform state, plan changes, and execute them through governed interfaces, always subject to the same least-privilege, fail-closed, and auditable controls that constrain every other actor in the system.

1.1 Scope and contributions

This paper makes three contributions:

1

A **taxonomy** of administrative domains in an enterprise AI platform, and the agentic capabilities appropriate to each (Section 3).

2

A **reference model** for how Administration Agents operate over a real platform's governance surface: identity, knowledge, agents, usage, proactive agent synthesis, and compliance (Sections 4 to 9).

3

A set of **safety principles** (fail-closed access, least privilege, human-in-the-loop for mutating actions, and pseudonymised immutable audit) that make autonomous administration trustworthy (Section 10).

02

Platform background

To ground the discussion, we summarise the reference platform at a conceptual level. Corporate GPT is deployed as a self-contained system inside the customer's boundary. A browser-based experience (a chat surface and an administration console) sits behind an edge tier that terminates TLS and enforces rate limits. An application tier serves the API, authenticates every request, and routes conversations to an agent orchestrator. Knowledge is ingested by a pool of asynchronous workers and stored in a vector database that supports

hybrid (semantic plus keyword) search, alongside a relational store for users, sessions, access-control records and configuration, and an in-memory store for coordination and caching. All model inference (chat, embeddings, vision, and speech) can run on self-hosted engines so that no data leaves the boundary, with optional managed model providers available where policy allows. Every model call is traceable through an in-boundary observability collector.

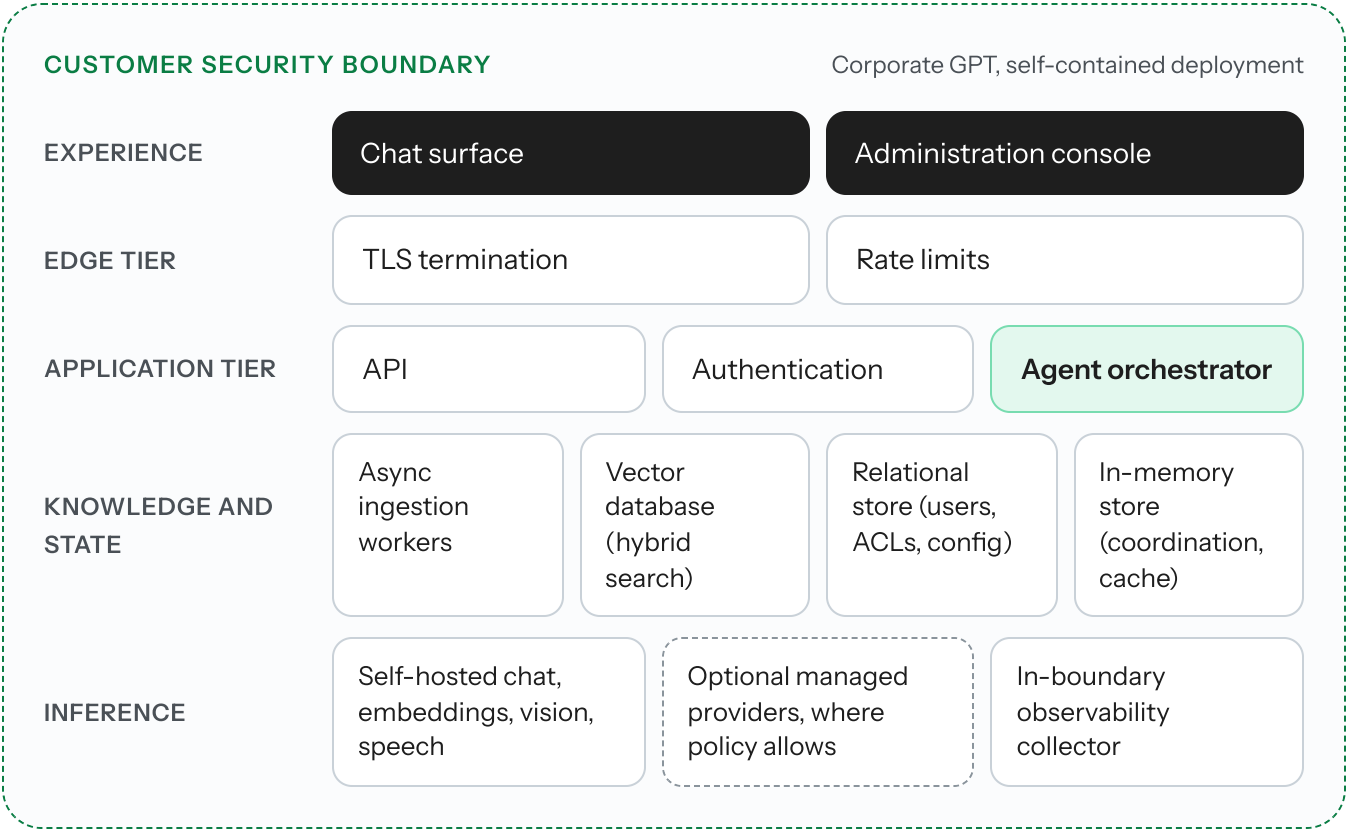


Figure 1. The reference platform at a conceptual level. Every tier runs inside the customer's boundary.

Two aspects of the platform matter most for this paper:

The agent ecosystem

The platform ships built-in agents (for document retrieval, web search, structured-database querying, and connector-backed systems such as issue trackers and support desks) and, crucially, an **Agent Builder** that lets authorised users compose new agents and multi-step workflows from a curated set of tools and connectors. Agents can be composed into trees, published as revisions, and shared with individuals or groups.

The governance surface

Underneath the ecosystem sits a rich control plane: role-based access control, per-document and per-memory access lists, connector kill-switches and scoping, credential vaulting, usage metering, privacy detection and redaction, and multiple immutable audit trails. This control plane is the substrate on which Administration Agents act.

03

What is an Administration Agent?

We define an Administration Agent as an agent whose objective is to observe, plan, and effect change on the platform's governance surface, rather than to answer end-user questions. It differs from an end-user agent in three ways: its tools act on platform state (users, permissions, connectors, agents, retention policy) rather than on knowledge; its privileges are elevated but tightly scoped and always attributed to a responsible principal; and its actions are, by default, proposed for human approval before they mutate anything consequential.

Administration Agents span six domains, summarised in Table 1. They range from fully autonomous background functions (for example, keeping directory groups synchronised on every login) through advisory agents that plan a change and present it for approval (for example, a knowledge-merge that would widen access), to interactive copilots that let an administrator express intent in natural language and translate it into governed operations, and, at the most proactive end, agents that notice a user's recurring routine and offer to automate it (Section 8).

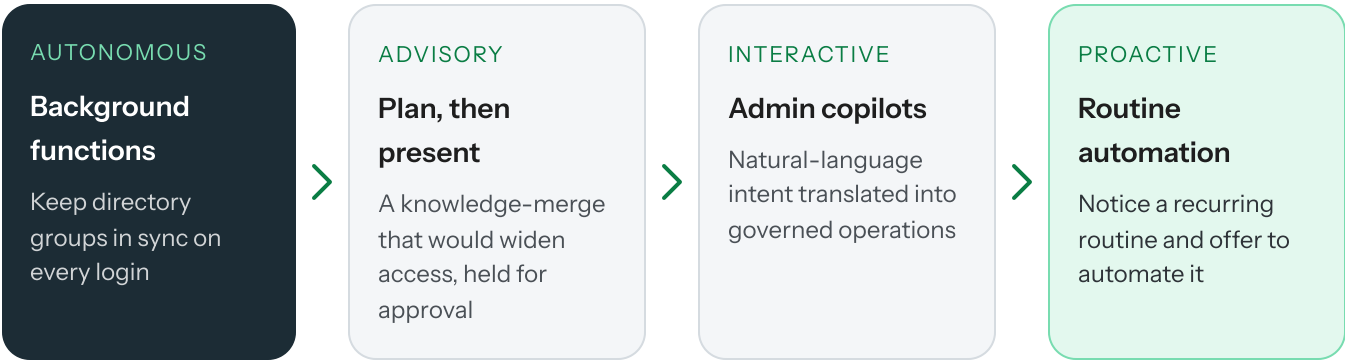


Figure 2. The autonomy spectrum of Administration Agents.

Domain	Administrative concern	Autonomy
Identity and access	Provisioning, roles, groups, directory sync, sign-on	Auto / Advisory
Knowledge and connectors	Onboarding sources, scoping, overlap and re-permissioning	Advisory
Agent governance	Capability palette, validation, publishing, sharing	Advisory
Usage and cost	Metering, quotas, rate limits, model and connector oversight	Auto / Interactive
Proactive synthesis	Pattern mining, drafting and proposing agents and workflows	Proactive / Opt-in
Compliance and audit	Privacy detection, data-subject rights, audit and retention	Advisory / Interactive

Table 1. The six administrative domains and the level of agent autonomy each typically warrants.

04

Identity and access administration

Identity is the foundation of every other control. The platform recognises a graded set of roles, from limited and basic users, through curators and global curators who steward shared knowledge and agents, to administrators who govern the system, and organises people into

two kinds of collections: administrator-managed access groups and user-created projects. Access groups can be hierarchical, which lets visibility flow down an organisational tree.

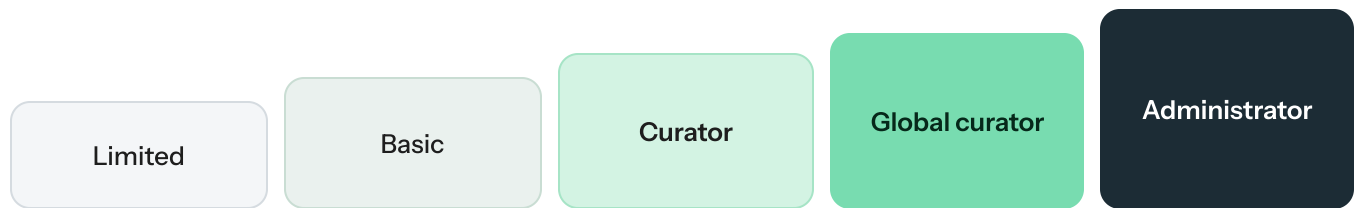


Figure 3. Graded roles, from users who consume, through curators who steward shared knowledge and agents, to administrators who govern the system.

4.1 Federated sign-on and just-in-time provisioning

The platform authenticates against an enterprise identity provider (OIDC-based single sign-on) and against on-premises directory services (LDAP bind), in addition to local accounts. New users are provisioned just in time on their first successful sign-on, so there is no separate account-creation backlog. The right administrative posture here is a background agent: on each login it reconciles the user's directory group memberships against the platform's access groups, adding and revoking membership so that entitlements never drift from the source of truth. Directory-synced groups are treated as read-only inside the console, and name collisions with manually managed groups are refused rather than silently merged: an example of the fail-closed stance that recurs throughout.

4.2 Guarding the front door

Sign-in itself is an administrative concern. Brute-force resistance is enforced with layered rate limiting, per source address and per account, backed by the coordination store, and it fails closed if that store is unreachable. Password changes are checked against a configurable policy and against reuse. An Administration Agent overseeing identity does not weaken these controls; it operates within them, and can surface anomalies (unusual login velocity, a spike in denials) as advisories for a human administrator.

4.3 Role and membership changes as governed actions

Elevating a user's role, moving someone between groups, or granting a connector to a person are exactly the kind of high-consequence changes that suit an advisory agent: the agent interprets the request ("give the new analyst the same access as the data team"), computes the concrete set of grants and revocations, presents the delta for approval, and only then executes. Every such change is written to an immutable audit trail (Section 9).

Interpret request



Compute grants and revocations



Approve delta



Execute



Immutable audit

05

Knowledge and connector administration

An enterprise AI platform is only as good as the knowledge behind it, and onboarding that knowledge safely is a substantial administrative task. Sources range from document repositories and network file shares to structured databases, issue trackers, support desks, and collaboration suites. Each source is exposed through a connector.

5.1 Access follows the document

Access control is enforced at the level of the individual retrievable chunk. Every ingested item carries an access-control stamp (who owns it, which users and groups may see it, and whether it is public to everyone or to privileged roles) and retrieval filters results against the asking user's identity and group hierarchy before anything reaches the model. Visibility is permissive within a document's stamp (any satisfied arm grants access) but strictly denied on error, so a failure never widens exposure. Because visibility can be inherited down a group hierarchy, an administrator can reason about access at the level of the organisation chart rather than per person.

ACCESS-CONTROL STAMP, PER CHUNK

Owner	Who owns it
-------	-------------

Users and groups	Who may see it
------------------	----------------

Public flags	Everyone, or privileged roles
--------------	-------------------------------

Any satisfied arm grants access

Permissive within the stamp, inherited down the group hierarchy.

Errors deny

A failure never widens exposure.

Figure 4. Chunk-level access control: retrieval filters on the stamp before anything reaches the model.

5.2 Connector scoping and kill-switches

Each connector type carries an org-wide enable switch and an access scope: available to everyone, or restricted to named users, named groups, or privileged roles. This gives administrators a coarse but instantaneous lever: disabling a connector type takes it offline everywhere without touching any agent's configuration. An Administration Agent can use these levers surgically (for instance, quarantining a source that has begun returning suspect content) while a human retains the final say.

5.3 Safe onboarding and re-permissioning

The most error-prone knowledge operation is onboarding a source that overlaps one already present: a folder nested inside an already-indexed folder, for example. Here the platform's design lends itself naturally to an advisory agent. Before any change is committed, the system can produce a dry-run plan: which records would move, which documents would be re-permissioned, and, most importantly, whether the merge would widen, narrow, or orphan access for any content. Presenting this access delta for explicit approval, rather than silently applying it, turns a dangerous bulk operation into a reviewable one. Conflicting registrations are serialised so that two administrators (or two agents) cannot race.

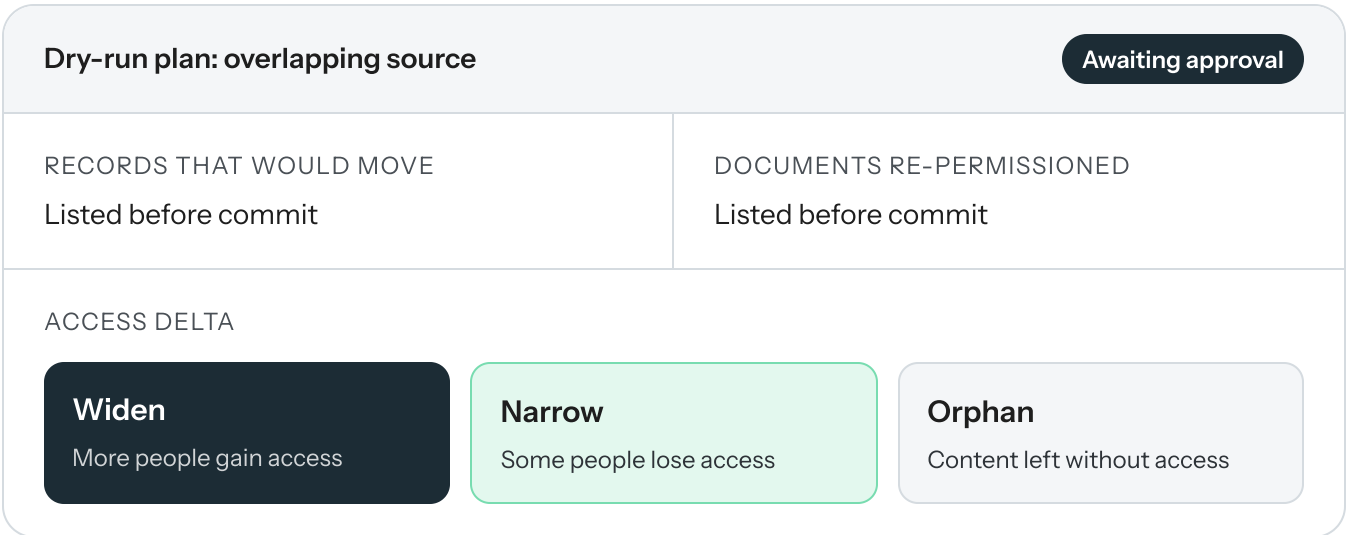


Figure 5. A dangerous bulk operation becomes a reviewable plan.

Every connector, credential, and scope mutation is recorded in a dedicated audit trail, with secrets scrubbed from the record before it is written: any field whose name suggests a key, token, password, or credential is dropped.

Agent governance

The Agent Builder is the platform's most powerful (and therefore most governance-sensitive) feature, because it lets people create new autonomous actors. Administering the agents themselves is a first-class domain.

6.1 The capability palette as a policy boundary

Users do not wire agents to arbitrary code. They select from a server-owned capability palette: a curated allow-list of tools (document retrieval, web search, chart and document generation, and so on) and connectors. Risky capabilities are feature-gated and can be withheld entirely; sensitive connectors require specific roles. The palette is, in effect, the policy boundary that an administrator draws around what any user-built agent is permitted to do. Curating this palette (deciding which capabilities are available to whom) is an administrative act well suited to an advisory agent that can reason about the blast radius of enabling a capability.

6.2 Validation, publishing, and sharing

Before a composed agent can run, it is validated: references to child agents and skills must resolve, composition trees are checked for cycles and bounded in depth and size, and any parameters bound to connectors are screened for injection and traversal attempts. Agents are versioned through a draft-then-publish workflow, and a published agent that other agents depend on cannot simply be deleted out from under them. Sharing is governed by graded access levels (view, run, edit, and the right to share) granted to individuals or groups. Built-in agents are read-only and cannot be mutated.



Graded sharing levels, granted to individuals or groups.

6.3 Guardrails at run time

When any agent runs, built-in or user-composed, it inherits a common set of guardrails. An Administration Agent governing this domain enforces these defaults and flags any agent whose configuration would weaken them.

Tool-call rate ceilings

Every run is bounded in how many tool calls it may make.

Human-in-the-loop gate

A confirmation before any mutating external action, such as writing to a connected drive.

Untrusted content stays data

All fetched web and third-party content is treated as data rather than as instructions, to blunt prompt injection.

Per-run credentials

Credentials are resolved per run under the runner's own identity, never embedded in an agent's definition, so an agent can never carry privilege it was not granted.

07

Usage, cost, and operational oversight

Once a platform is in daily use, administrators need to see and shape how it is consumed. The platform meters connector usage at the granularity of a single tool call, recording the connector, the person, the session, and the outcome (success, denial, or error), and it records the lifecycle of every model and connector configuration (creation, activation, rotation, deactivation). Importantly, this telemetry is written on a best-effort, non-blocking path so that oversight never slows the answer.

PER TOOL CALL

Connector Person Session

Outcome: success, denial, error

CONFIGURATION LIFECYCLE

Creation › Activation › Rotation ›

Deactivation

Figure 6. The metering that serves as the Administration Agent's sensor array.

This metering is the natural sensor array for an Administration Agent. Because connector types, individual credentials, and model configurations can all be toggled through configuration rather than code, an oversight agent can act on what it observes: throttle a

runaway integration, raise a quota for a team that has outgrown its allocation, or recommend deactivating a model that is no longer cost-effective. As always, consequential changes are proposed for approval and recorded.

The same telemetry supports a more ambitious, generative form of administration, described next.

08

Proactive agent synthesis from usage patterns

The domains so far describe an Administration Agent that governs what already exists. The most distinctive capability of the platform points the other way: an Administration Agent that observes how people work and builds them something better. Because every meaningful interaction is metered (Section 7) and because the Agent Builder can compose new agents and workflows programmatically (Section 6), the platform can close the loop between the two, turning a recurring manual routine into a one-click agent, offered to the very person who would benefit.

8.1 From repetition to recommendation

Consider a project manager who, every Friday afternoon, opens the issue tracker, pulls all open tickets in their project, sorts them by priority, and drafts a plan for the following week. To the person it feels like "just my job"; to the platform it is a signature: the same sequence of tool calls, over the same source, on the same cadence, producing the same shape of output. A pattern-mining Administration Agent watches the anonymised stream of a user's own activity for exactly these signatures: a recurring set of steps, correlated in time and target, that has repeated often enough to be worth automating.

When it finds one, it does not silently act. It drafts a candidate agent or workflow that reproduces the routine (here, a small workflow that queries the open tickets, ranks them, and assembles the priority list) and surfaces a plain-language proposal to the user:

"Every Friday afternoon you pull all open tickets in your project and put together a priority list for the coming week. I've drafted an agent that does this end-to-end. [Want to try it?](#)"

Run once

Edit in Agent Builder

Adopt weekly

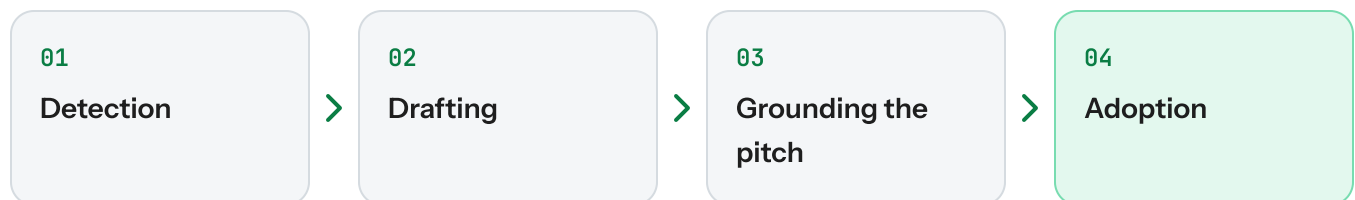
Dismiss

Figure 7. The proposal, in the user's own terms. The recommendation is an offer, never an imposition.

The user can run it once to see the result, edit it in the Agent Builder, adopt it (optionally on the same weekly schedule), or dismiss it. Only if they adopt it does anything become permanent. The recommendation is an offer, never an imposition.

8.2 How a proposal is formed

Synthesis reuses the machinery already described rather than inventing a parallel path:



- **Detection.** The agent mines the user's own usage signatures for recurring, multi-step routines, clustering by the tools invoked, the sources touched, the temporal cadence, and the shape of the output, and scores each candidate by how often and how regularly it recurs, so that noise and one-off explorations are filtered out.
- **Drafting.** It expresses the routine as a draft agent or workflow composed only from capabilities the user is already entitled to use. The proposal is created as an unpublished draft, exactly as if the user had built it by hand.
- **Grounding the pitch.** The natural-language proposal names the observed pattern in the user's own terms and links to the draft, so the recommendation is explainable: the user can see why it was suggested, not just what was suggested.

- **Adoption.** Accepting the offer runs the draft through the same validation, publishing, and (optional) scheduling path as any hand-built agent. The user owns the result and can revise or delete it at will.

8.3 Why this is safe by construction

Proactive synthesis is powerful precisely because it inherits every guardrail already in place, rather than bypassing them:

No new privilege

A proposed agent can only draw on capabilities and sources the user already has. The recommendation cannot widen access, and running the draft is subject to the same per-chunk access control as any other query.

Opt-in, never automatic

The routine is proposed, not deployed. Nothing runs on the user's behalf, and no schedule is created, until the user explicitly adopts it: the human-in-the-loop principle applied to creation itself.

Privacy-respecting detection

Mining operates over a user's own activity through the same best-effort, pseudonymised telemetry used for oversight. It reads the shape of behaviour (which tools, which sources, how often), not the content of anyone's data.

Fully audited

Every proposal offered, and every adoption or dismissal, is written to the immutable audit trail, so the organisation can see what the platform suggested and what its people chose to accept.

The result reframes the platform from a tool people operate into a colleague that notices the toil in their week and quietly offers to take it off their hands, while leaving them entirely in control of whether it does.

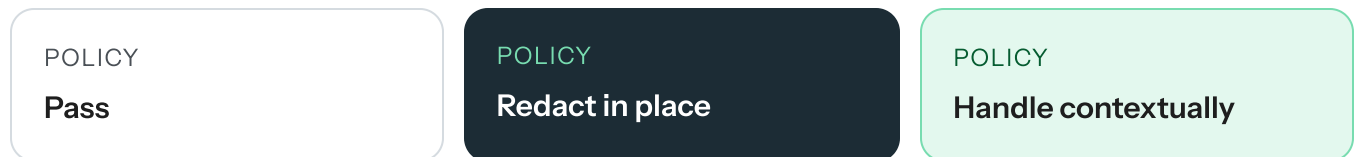
09

Compliance, privacy, and audit

For regulated enterprises, compliance is not a feature bolted on at the end; it is a property the platform must hold continuously. This is the domain where Administration Agents deliver the clearest return, because the work is high-volume, rule-driven, and unforgiving of lapses.

9.1 Privacy detection and redaction

As knowledge is ingested, it passes through a privacy pipeline built on established detection technology (Presidio with spaCy language models), which recognises a broad range of personal and regulated entities: national identifiers, payment details, contact information, health and financial terms, and more. Each entity type is handled by policy: some pass, some are redacted in place, and some (such as email addresses) are handled contextually, shielding incidental identifiers while never masking high-risk ones. Detection is designed to catch entities that straddle chunk boundaries, so nothing slips through the seams.



9.2 Traceability without exposure

Every detection is logged so that a redaction can be correlated later, but the log stores a keyed, one-way hash of the entity rather than the value itself. Hashing is tenant-scoped, so a breach of one tenant's records cannot be used to enumerate another's, and it survives key rotation through versioning. This is the pattern that lets an Administration Agent answer "where does this person's data appear?" without ever holding the data in the clear.

9.3 Data-subject rights

The platform treats the right to access and the right to erasure as first-class operations. A user's own memory and data can be exported under access-scoped controls to satisfy a subject-access request. Erasure ("forget") cascades across both the authoritative store and every derived index, and, for git-curated shared knowledge, is performed as a tracked, reviewable change. These are precisely the workflows an interactive compliance agent can drive: interpret the request, locate every affected artefact through the audit and hashing machinery, present the scope for sign-off, and execute the cascade.

9.4 Immutable, pseudonymised audit

Every administrative domain writes to an append-only audit trail. Actor identities are recorded as stable one-way pseudonyms: an auditor can tell that the same person performed two actions without the log itself revealing who they are, and secret material is never written. Separate trails cover identity and access changes, connector and credential

mutations, model configuration, agent creation and sharing, privacy detections, and memory governance. Together they give an Administration Agent both its evidence base (what happened) and its accountability record (what the agent itself did).

SEPARATE APPEND-ONLY TRAILS

Identity and access changes

Connector and credential mutations

Model configuration

Agent creation and sharing

Privacy detections

Memory governance

— 10

Safety principles for autonomous administration

Delegating administration to agents is only responsible if the delegation cannot be turned against the organisation. The reference platform embodies a consistent set of principles that make this delegation safe; they are, in our view, the minimum bar for any Administration Agent.

01 Fail closed

Every access decision denies on error. Ambiguity, missing identity, or an unreachable dependency results in less access, never more. Agents inherit this stance; an agent that cannot confirm it is allowed to act, does not act.

02 Least privilege, no escalation

Credentials are resolved at run time under the acting principal's identity and are never embedded in a definition. An agent cannot accumulate or lend privilege it was not explicitly granted.

03 Human in the loop for mutation

Read and analysis operations can be autonomous; consequential changes (widening access, altering roles, deleting data, writing to external systems) are proposed as reviewable plans and gated on human approval by default.

04 Everything is attributed and audited

Every action an Administration Agent takes is attributed to a responsible principal and written to an immutable, pseudonymised trail, with secrets scrubbed. Autonomy without accountability is not permitted.

05 Curated capability surfaces

Agents act only through server-owned allow-lists of tools and connectors, screened for injection and abuse. The platform, not the prompt, decides what is possible.

06 Data minimisation by construction

Sensitive values are detected, redacted, and referenced by keyed hash rather than by value, so administrative reasoning about personal data does not require handling it in the clear.

11

Reference architecture in practice

Bringing the domains together, an Administration Agent operates in a loop that mirrors how the platform's own agents work, but pointed inward at the control plane:

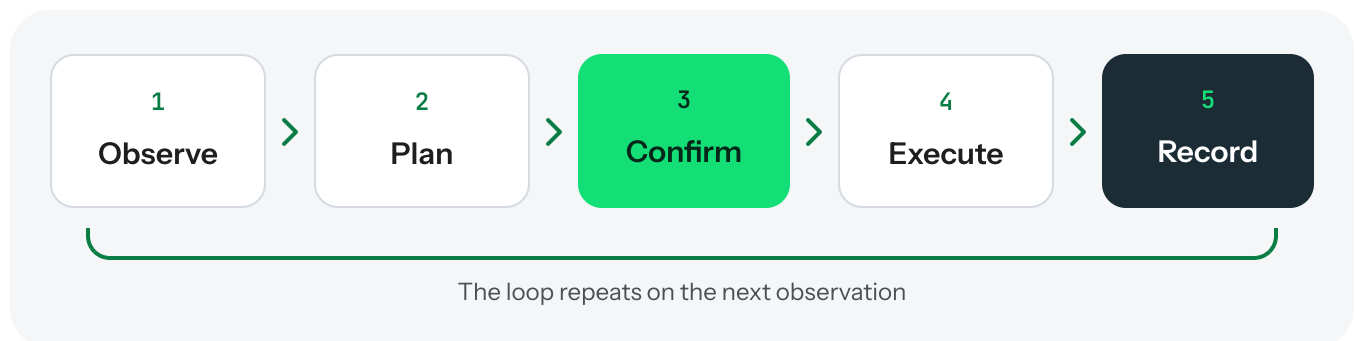


Figure 8. The Administration Agent loop, pointed inward at the control plane. Approval in step 3 is itself an audited event.

1. **Observe.** It reads platform state through governed, read-scoped interfaces: the usage meters, the audit trails, the access catalogues, the connector and agent registries.

2. **Plan.** It composes a concrete change set and, for anything consequential, computes the delta and its blast radius; for example, the exact widening or narrowing of access a proposed merge would cause.
3. **Confirm.** It presents the plan to a human administrator, whose approval is itself an audited event.
4. **Execute.** It effects the change through the same governed operations a human would use (toggles, scopes, grants, retention actions), never through a privileged back door.
5. **Record.** It writes the outcome to the appropriate audit trail, attributed and pseudonymised.

Notably, the same Agent Builder that serves end users can itself be used to compose Administration Agents, subject to the elevated capability palette that administrative work requires. Governance and capability are thus expressed in one consistent model, rather than as a separate, less-scrutinised admin toolchain.

— 12

Conclusion and future directions

Enterprise AI succeeds or fails on governance. The platforms that will move beyond pilots are the ones that can be administered at the scale of an organisation, with entitlements that never drift, knowledge that is onboarded without accidental exposure, agents that are curated rather than uncontrolled, usage that is visible and shapeable, and compliance that holds continuously. Administration Agents are how that scale is reached. By pointing the agentic paradigm inward (at identity, knowledge, agents, usage, and compliance) and by constraining it with fail-closed access, least privilege, human-in-the-loop mutation, and immutable pseudonymised audit, an enterprise can delegate the administrative burden without ceding control. And by turning that same inward gaze onto how people actually work, the platform can go further still, noticing a recurring routine and proactively offering a purpose-built agent to retire it, so that automation finds the user rather than waiting to be built.

The reference platform shows that the necessary substrate (graded roles, chunk-level access lists, connector kill-switches, dry-run access deltas, curated capability palettes, and layered audit trails) can be built today. Future work lies in raising the autonomy ceiling safely:

NEXT

Policy-as-code that
Administration Agents can
propose and reason over.

NEXT

Continuous entitlement
review that flags access no
longer justified by
behaviour.

NEXT

Cross-domain agents that
connect a compliance
signal to an access change
to a usage anomaly in a
single, auditable line of
reasoning.

THE DIRECTION IS CLEAR

**The enterprise AI platform of record
will administer itself, **under human
authority**, one governed action at a time.**