empowerID

WHITEPAPER · AGENT TEAMS

From agent pilots to governed operations

How EmpowerID Agent Teams operates AI agents as durable, chartered teams on the Identity Fabric — explicit authority, deterministic run lifecycle, autonomy matched to consequence, human confirmation that stops execution, and evidence from charter to result.

July 2026 For security & identity leaders, platform owners, and operations

EXECUTIVE SUMMARY

Agents now monitor conditions, plan multi-step tasks, coordinate with other agents, and propose or execute changes across business systems. That work raises questions an agent builder cannot answer: who owns the agent after deployment, what is the team chartered to do, whose authority does it exercise, which steps may run under policy and which must wait for a human, who can stop it — and what evidence connects charter, decision, approval, and result.

EmpowerID Agent Teams is the governed operations application and runtime on the EmpowerID Identity Fabric. Where agent governance controls what each actor may do, Agent Teams organizes the actors into the unit that does the work: durable teams of registered agents in named roles, with

charters and graph-backed authority, each chartered to own a recurring outcome. Heartbeats initiate proactive work; runs and stages give model reasoning a deterministic lifecycle; Contract-Driven Autonomy turns consequential steps into structured decisions — execute, request input, wait for confirmation, or fail; Agent Teams Studio gives operators fleet, run, approval, and stop controls. The surrounding fabric separately governs identity, delegation, model calls, tool calls, execution, credentials, and evidence. One principle organizes the architecture: the model contributes intelligence; the platform owns the operating lifecycle.

01 — THE PROBLEM

A prompt is not an operating model

A prototype agent is a model, a system prompt, tools, and a loop. Production work adds durable schedules, ownership, delegated authority, run state and recovery, approval queues, credential handling, stop controls, and evidence — concerns that do not fit inside the prompt. When they stay in application code or model instructions, each agent becomes its own miniature control plane, and three failures follow:

POLICY DRIFT

Each agent carries its own copy of the rules. Approvals become advisory, agent identity is confused with an API key, and no two teams enforce the same boundary the same way.

INVISIBLE STATE

Operators cannot see whether work is active, waiting, failed, or silently stuck — the only record is a chat transcript that omits policy decisions, retries, and lifecycle state.

NO OFF SWITCH

Stopping an agent means finding the right deployment, process, secret, or queue. There is no authorization-protected pause, suspend, or kill-switch — and no record of who stopped what, when.

The issue is operability as much as security. Gartner reported in September 2025 that 75% of surveyed IT application leaders were piloting or deploying some form of AI agent, while only 15% were considering fully autonomous ones — governance, trust, and agent sprawl remained the barriers. Its 2026 sprawl guidance calls for centralized inventory, explicit identity and lifecycle, behavior monitoring, and remediation. Inventory answers which agents exist. Operations answers what they are doing now, what they are waiting for, which authority they are using, and whether they should

continue. An enterprise needs both: an agent system of record — in the EmpowerID portfolio, Agent Hub — and an agent operating layer. Agent Teams is the second, and the difference from its siblings is the shape of the work itself:

Surface	Work starts when	Control flow	The human is
Agent Hub	An admin registers an identity	—	Configuring ownership and delegation
AI Assistant	A user types in chat	Model-driven, turn by turn	In the loop on every message
Agent Teams	A heartbeat fires or a brief arrives	Platform-owned run and stage lifecycle	A supervisor at the mutation gates

> The near-term opportunity is bounded agency, not unrestricted autonomy. Specific workflows where the organization can define authority, controls, owners, and value — with autonomy increased only where evidence supports it.

02 — THE CATEGORY

Every platform is announcing an agent control plane

By mid-2026 the major enterprise software vendors ship or have announced agent control planes: centralized observation and governance, multi-agent orchestration, governed catalogs, scheduling, runtime monitoring, and revocation. Billion-dollar acquisitions in agentic automation reinforce the direction. Visual agent construction, role-based coordination, templates, persistent workspaces, triggers, approvals, and activity dashboards are becoming expected capabilities — a buyer should assume them. The differentiating question sits underneath:

Expected control-plane capability	Governed-operations requirement
Multi-agent canvas and templates	Durable teams with charters and graph-backed authority, distinct from any single run
Agent inventory and catalog	Registered identities with accountable owners and scoped, revocable delegation
Scheduling and triggers	Heartbeats claimed under durable leases, producing explicit run and stage state
Human-approval steps	Confirmation that stops execution at the runtime boundary — with target, authority, and expiry
Activity traces and dashboards	Evidence linking charter, delegation, policy decision, approval, execution, and result
Model and tool integration	Model and tool calls governed by specialized enforcement points on a shared policy plane

Can the agent operating layer participate in the organization's identity, delegation, authorization, execution, credential, and evidence systems deeply enough to carry consequential work? That is the Identity Fabric question, and it is where Agent Teams is built to differ.

03 — THE MODEL

A team is a charter, not a group chat

Governing a single agent answers what one actor may do. An agent team is a different object: a small, durable working unit — several registered agents in named roles, usually with humans in the loop — chartered to own one recurring outcome and hand work between roles to deliver it. A reliability team might pair a monitoring agent that sweeps systems on a heartbeat with an analyst agent that classifies exceptions and a communicator that drafts the escalation a human approves. The work is assigned to the team the way an organization assigns work to a department: the unit owns the outcome, the roles divide the labor, and the handoffs are recorded.

That group becomes an *enterprise* team when its authority and operating contract are explicit. The charter defines what outcome the team is responsible for, who may submit work, which agents may fill each role, which systems, models, tools, and data are in scope, which actions are read-only, advisory,

approval-bound, or autonomous, what cadence initiates work, what evidence is retained, and who owns exceptions and may stop the team. Team membership participates in authorization — it is graph-backed authority, not a UI assignment. Three objects stay distinct:

1 · Team

Durable. Holds the charter, role bindings, authority and approval model, capacity, and lifecycle state. Standing teams return to ready after a sweep; project teams complete or archive.

2 · Run

One brief or operational cycle — started by a heartbeat, approved event, or submitted brief. Concurrent runs are capacity-managed, never collapsed into one shared chat history.

3 · Stage

An attempt within a run: input, model contribution, artifact, and a recorded handoff or terminal state — the unit where policy, approval, and evidence attach.

> **Why the separation matters.** A common design failure treats team, project, conversation, and execution as one object. Then state lives in the transcript, recovery means re-reading the chat, and authority means whatever the prompt last said.

04 — PROPORTIONAL AUTONOMY

Autonomy matched to consequence

Gartner advises against applying uniform governance across agents with different autonomy levels: observation, advice, action with approval, and autonomous action each need different controls, with stronger monitoring, rollback, and circuit breakers as autonomy rises. Agent Teams applies this per step, not per agent — the same team can read freely, draft with review, and stop for confirmation on the one step that changes enterprise state:

Operating mode	Typical access	Human role	Required controls
Observe	Read approved data	Reviews output	Identity, scoped access, logging
Advise	Read and draft	Decides and acts	Output review, quality evaluation, attribution
Act with approval	Prepare a write or action	Confirms each consequential step	Structured preview, authorization, audit, incident response
Bounded autonomy	Execute within defined constraints	Reviews exceptions and outcomes	Continuous monitoring, circuit breaker, stop/rollback, ownership

The runtime mechanism is Contract-Driven Autonomy: every proposed step resolves to a structured outcome —

`EXECUTE`	policy and required context permit the step — it runs through the governed execution path
`NEED_*`	a missing parameter, clarification, or resolution is required before the step can proceed
`REQUIRE_WAITING`	the proposed step stops for human confirmation — a durable state, not a suggestion in the prompt
`FAIL`	policy or runtime conditions block the step, with a visible, machine-readable reason

> **Approval that means something.** A confirmation task shows the proposed action, the target system and object, before-and-after context, why confirmation is required, the requesting agent and team, the authority under which it will execute, and expiry. Approvals expire rather than silently succeed. The goal is not to maximize approval count — it is to place human accountability at the decisions where it changes risk.

The execution boundary carries one more control most agent platforms lack: a side-effect ledger. Before an external mutation is attempted, a durable reservation with an idempotency key is written — and the step fails closed if it cannot be. Retries never duplicate a send, drift between the reservation

and the observed result is detectable, and when an external system's state is unknown after a fault, the record says so explicitly instead of guessing. Most platforms log actions after the fact; the ledger constrains them before.

05 — DIVISION OF LABOR

Deterministic control around probabilistic reasoning

Models are valuable because they interpret ambiguity, summarize context, propose plans, classify, and draft. Those strengths do not make them good owners of operational lifecycle state. Agent Teams draws the line explicitly:

The model may contribute	The platform owns
Interpret a heartbeat result; summarize changes or exceptions	Schedule and lease state; run and stage identifiers
Classify severity or intent; propose a plan	Inbox/outbox contracts; schema validation; role binding and team authority
Draft a communication; produce a stage artifact	Retry and capacity rules; transitions and handoffs; confirmation status
Select among policy-permitted next-step patterns	Terminal state; receipt and event emission; pause, suspend, and kill-switch

The line was drawn from production experience, not preference. Early designs that asked a model to drive a multi-step inbox lifecycle failed in familiar ways — identifiers misrouted, work abandoned after the first step — and prompt fixes did not repair them, because a probabilistic model is not a durable state machine. The corrected architecture inverts the responsibility: a deterministic graph executor owns claim, read, write, submit, and complete, and the model sits inside one bounded step, returning typed artifact content — never deciding what runs next.

This separation makes the system testable and auditable: model behavior can evolve without changing the meaning of waiting, approved, failed, complete, or stopped. The claim is deliberately narrow — deterministic lifecycle does not guarantee that model-generated content is correct. Organizations still need model evaluation, output validation, red teaming, and human judgment appropriate to the use case. What determinism prevents is a probabilistic model becoming the sole authority over operational state and consequential execution.

06 — WHAT MAKES IT DIFFERENT

The operate layer of the Identity Fabric

Agent Teams does not re-implement governance inside every team prompt. It is the operating layer above specialized fabric services: Agent Hub registers and inventories the actors; identity and delegation establish whose authority an agent exercises; the shared PDP decides; the LLM Gateway governs model calls; the MCP Gateway governs tool calls; the Orchestration Service executes native enterprise work under Contract-Driven Autonomy; evidence services record and correlate. NIST's 2026 agent-standards initiative calls for exactly this separation of agent identity, authorization, auditing, and non-repudiation concerns:

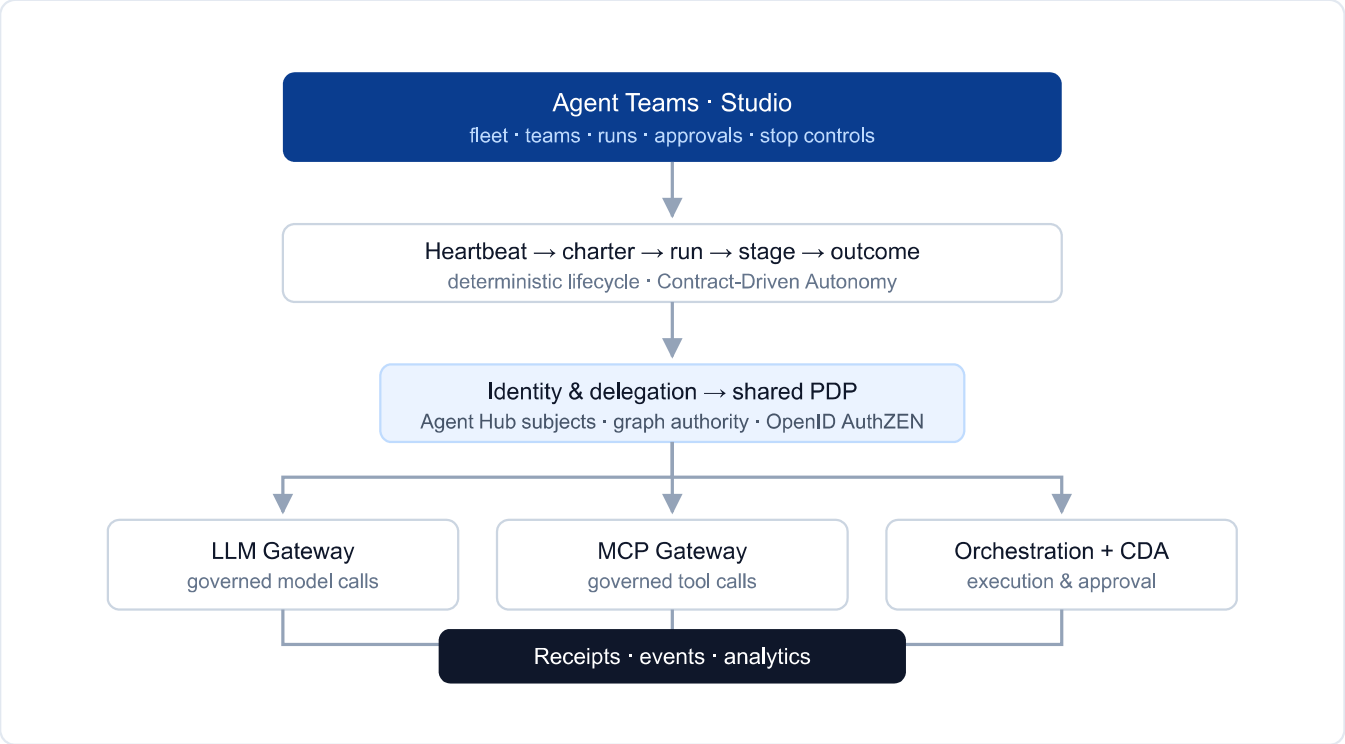

Figure 1 — Agent Teams operates; the fabric's specialized services identify, decide, enforce, execute, and remember.

07 — HOW IT WORKS

From heartbeat to evidence

Every governed cycle follows the same path, whether it began as a schedule, an approved event, or a submitted brief:

1

Initiate bounded work

A durable heartbeat, approved event, brief, or template starts a run — tied to a specific team and operating contract, never a generic agent with ambient access.

2

Claim and identify the run

The scheduler claims due work under a durable lease suitable for distributed execution; the runtime creates or resumes explicit run and stage state. Skip rules — active hours, budget exhaustion, delegation validity — are checked before any model is called, and each check can use its own model tier, so routine sweeps run on inexpensive models.

3

Resolve authority

Team, agent bindings, subject identity, active delegation, and policy context are resolved. An agent identity without the required delegation cannot proceed on the governed path — there is no fallback to a shared application identity.

4

Contribute intelligence within the stage

The model analyzes input, creates a structured artifact, proposes a plan, or drafts a response — with model access flowing through the LLM Gateway's own governed boundary.

5

Evaluate the proposed step

The runtime and policy layer resolve the step to execute, request input, wait for confirmation, or fail — a durable operational state, visible to operators.

6

Execute through the approved path

Tool use goes through the MCP Gateway; native enterprise work through the Orchestration Service. Downstream credentials are brokered by the execution path — never returned to the model or agent.

7

Record handoff and evidence

Stage transitions, artifacts, approval events, model and tool receipts, and operational telemetry update the governance timeline.

8

Reach an explicit outcome

The run completes, returns a standing team to ready, remains waiting, enters failure handling, pauses, or stops. Operators never infer state from the last chat message.

Figure 2 — The operating cycle on every governed run.

08 — OPERABILITY

Attention, stop, and recovery

The operating experience must tell a human where attention is needed — healthy, waiting for information, waiting for confirmation, degraded, failed with a visible reason, paused, or stopped. Three stop controls serve different needs, each authorization-protected and recorded in the operating timeline:

1 · Pause

Stops new scheduled work while preserving configuration — the team resumes where it left off.

2 · Suspend

An administrative or policy hold with a recorded reason — for review, audit, or investigation.

3 · Kill-switch

Stops an agent instance rapidly when risk or malfunction demands immediate action.

Failure semantics follow the fabric's discipline — each dependency has explicit, testable behavior:

`missing delegation`	fail or wait with structured advice — never fall back to a shared identity that expands authority	`PDP outage`	consequential work does not silently bypass authorization; last-known-good is explicit and bounded
`provider outage`	retry or failover only within model policy — availability never overrides constraints	`connector outage`	stage state is preserved, the dependency failure is surfaced, and retry or escalation follows policy
`approval timeout`	a waiting task expires or escalates by policy — silence never becomes approval	`worker offline`	device work stays unclaimed or visibly degraded — absence is never permission to bypass the local step

The last chip covers work that reaches beyond cloud APIs — a browser interaction, a local shell command, screen evidence, a device notification. Instead of a permanently privileged desktop agent inheriting the user's environment, the EmpowerID Local Worker is enrolled through a pairing flow and executes only supported connectors from signed execution envelopes sent by the control plane, returning results and evidence to the governed path. That does not make local automation risk-free; it puts a defined control boundary around the device step. Inbound channels get the mirror-image gate: a Telegram or email sender is paired and approved — pending, approved, rejected, expired — before an inbound message can start any tool execution, separately from the team's outbound contact policy.

09 — FOR THE SECURITY ORGANIZATION

A transcript is not a system of record

A chat transcript omits policy decisions, retries, tool execution, human approval, downstream results, and lifecycle state. The operational record Agent Teams assembles answers the reviewer's questions directly: which team and charter initiated the work, which registered agents filled the roles, whose delegated authority was used, which run and stage produced the proposed action, which policy decision or runtime outcome applied, whether confirmation was required, granted, rejected, or expired, which model and tool paths were used, what artifacts resulted, and whether the agent was paused, retried, failed, or stopped. Stage and handoff audit, approval state, operational events, and analytics combine with signed receipts from the covered model and tool paths.

> Stated boundaries. Evidence represents what was decided and what executed — not a signed receipt for every denied action or internal state transition. And Agent Teams does not claim automatic learning from outcomes, a connector marketplace, built-in anomaly detection, or fully autonomous customer communication as a default. It operates alongside Agent Hub, the LLM Gateway, and the MCP Gateway; it replaces none of them.

Five questions to ask any agent-operations platform

1. Is the team durable and separate from a run — with a charter that participates in authorization, or just a canvas grouping agents in a UI?
2. Does every agent have a distinct identity, an accountable owner, and scoped, time-bound, revocable delegation — checked at run time?
3. Does human approval stop execution at the runtime boundary — with target, effect, authority, and expiry — or is it a message the agent may route around?
4. Can the system recover a run without reconstructing state from a chat transcript — and can an authorized operator pause, suspend, or kill an agent, with the action recorded?
5. Are model and tool calls governed by specialized enforcement points on a shared policy plane — or does each team prompt carry its own copy of the rules?

These questions separate a multi-agent canvas from an operating layer. Agent Teams is the operate component of EmpowerID Agent Governance & Execution — the same identity, authorization, credential, and evidence services that govern the rest of the enterprise, applied to proactive, durable agent work.

KEY TAKEAWAYS

The near-term enterprise opportunity is bounded agency: defined workflows with owners, authority, checkpoints, and measurable value — not unrestricted autonomy.

A team is the unit of work: durable charter, named roles, graph-backed authority — distinct from any run, any conversation, any prompt.

Consequential steps become durable states — execute, need input, wait for confirmation, fail — enforced by the runtime, not suggested in the prompt.

The model contributes intelligence. The platform owns the operating lifecycle. The fabric establishes the authority. The record survives the transcript.

See a team run from heartbeat to evidence

Watch a standing team wake on schedule, stage its work, stop for a human confirmation on the consequential step, execute through the governed path, and land the whole cycle in the operating timeline.

Request a demo

info@empowerID.com · empowerid.com

empowerID