

EREZAGENT

AI COMPANY OPERATING SYSTEM

Technical Capabilities & Platform Features

How ErezAgent turns goals, decisions and business requirements into governed execution.

THE PURPOSE OF THIS GUIDE

This document explains what the ErezAgent platform actually does at product and technical level. It is intentionally more detailed than the Commercial Overview and more accessible than the internal Architecture Reference. It describes implemented behavior in product terms, including the controls and boundaries that make the behavior reliable.

Technical Product Guide

Baseline: v4.94.0 • Migration head 564 • October 2026

One Operating System, Several Ways to Enter

ErezAgent is not a collection of unrelated AI tools. The same governed execution platform can begin with a goal, a mission, a customer requirement, or an existing business and continue into real operational work.

A user can begin with an outcome such as “build a business that can make \$5,000 per month.” The Start experience shapes that intent into an Intake Brief containing a measurable goal, metric, target, timeframe, and scope. From there, the system can draft a company priority, venture proposal, or objective and send it to the Executive Board. Nothing becomes active simply because the AI suggested it; the user reviews and confirms the transition.

The same platform also accepts a more concrete mission, such as building a SaaS product, or customer requirements that need to be delivered as a solution. In an established company, Business Twin can ingest existing processes, documents, and operational material and propose a structured model of how the business works. These are different entry points into the same runtime, not separate products.

WHAT MAKES THE PLATFORM TECHNICAL RATHER THAN CONVERSATIONAL?

The important distinction is continuity. An answer can become a governed objective, mission, project, workflow, business record, customer action, build, deployment, or billing event. ErezAgent is designed to preserve context and control as work moves from reasoning into execution.

Starting point	What the platform constructs	What can follow
Goal or desired outcome	Intake Brief, measurable objective, evidence request	Board deliberation, venture proposal, missions and execution
Business or product idea	Validation, requirements, plan and capability needs	Build, launch, GTM, operations and measurement
Customer requirement	Traceable requirements and solution blueprint	Configuration, extension, custom build, verification and acceptance
Existing business	Evidence-backed Twin of processes, roles, records and workflows	Approved configuration, staff workspaces, portal and automation

Reasoning Is Separated From Authority

Every AI-driven action ultimately passes through the same execution and governance chain, regardless of whether it originated in chat, a mission, a scheduled job, a connector, or MCP.

The Executive Board can deliberate on an objective using multiple executive viewpoints and company context. An agent turn then assembles the agent persona, operating doctrine, venture context, mission context, and permitted tools. When an action is needed, the Skill Executor is the common execution seam: it validates permission and risk, runs the skill, records the invocation, captures cost, and returns a governed result.

This separation is deliberate. The language model can reason about what should happen, but it does not automatically gain permission to make that change. The execution layer decides whether an operation may run immediately, must wait for approval, is blocked by policy, or exceeds a spending or capability boundary.

Autonomy is bounded in code

The action registry assigns built-in actions an autonomy tier from T0 to T3. A tenant policy may make an action stricter, but cannot lower the mandatory approval floor of a T3 action. Per-action daily spending ceilings are checked at the same gate; exceeding the ceiling forces the action into approval rather than allowing silent overspend.

Operating Doctrine provides another control layer for communication tone, evidence expectations, publishing restrictions, and spending boundaries. Doctrine is resolved from the most specific scope first—mission, agent, department, venture, then company—but security floors and mandatory approval rules are structurally excluded from doctrine so they cannot be weakened by a convenient prompt or configuration change.

PRACTICAL EFFECT

A user can ask an agent to send, publish, delete, execute, or spend, but the request still has to survive capability checks, role permissions, action risk and autonomy tier, spending limits, the agent's own policy for the capabilities it governs, approval, and audit requirements before the side effect occurs.

Executive Board, Objectives and Measurable Value

The strategic layer is designed to make important decisions with explicit business context rather than treating every prompt as an isolated request.

For strategic objectives, the Board can combine Company Foundation, Venture Foundation, organizational context, historical lessons, and current operational evidence. Executive roles contribute independently; a challenge round can surface disagreement, and the final synthesis can include abstentions or requests for missing evidence rather than manufacturing certainty.

The result is not meant to remain as advice. Approved strategic direction can become a measurable objective, which can then drive missions and work. ErezAgent’s value model requires objectives, missions, automations, and resource allocations to identify what measurable value they intend to create, how the value will be observed, and which assumptions need to be tested.

That creates a closed loop between strategy and operations. Revenue, mission outcomes, customer health, operational events, and other real signals can feed later decisions, while institutional lessons and executive calibration improve future deliberation. The system is therefore designed to learn from what actually happened, not only from what agents previously wrote.

Technical element	Role in the system
Company Foundation	Company-wide identity, principles and active company priority supplied to strategic reasoning.
Venture Foundation	Venture-specific mission, market, business model and operating context.
Executive Board	Multi-role deliberation, challenge, synthesis, abstention and evidence acquisition.
Objective Center	Tracks metrics, targets, assumptions, progress and evidence around desired outcomes.
Institutional memory	Feeds lessons, decisions and relevant organizational history back into future work.

AI WORKFORCE

A Governed AI Organization, Not a Single General-Purpose Assistant

ErezAgent coordinates 46 specialized AI agents whose identity, tools, budgets, and permissions are controlled independently from the language model that happens to power a particular turn.

Core agents are shared templates, while each tenant can have its own overlay for enabled state, status, budget, default model, system-prompt override, department, and capability routing. This allows one installation to preserve a stable organizational model while giving each customer a different operating profile. A tenant can change how an agent behaves without editing the global template used by other tenants. Agent policies—allowed connectors, approval mode, and whether an agent may create missions or memories or run runbooks—are saved per workspace, so two tenants sharing the same built-in agent can hold different policies. Five capability flags (read CRM, create deals, send WhatsApp, spawn sub-agents, modify agents) are enforced at run time.

Agent identity is also separated from authority. An agent can have a persistent name, avatar, tone, voice, and language, but those persona properties do not expand what it is allowed to do. Tool access comes from granted skills and connector capabilities, and consequential changes to agent prompts, permissions, budgets, or models remain governed.

Agents can delegate work while preserving lineage and depth limits. Humans and AI agents can also own the same real work items, with governed reassignment between them. The result is an operating model in which AI workers participate in the same task and mission structures as people instead of living in a parallel chat-only universe.

WHY SPECIALIZATION MATTERS

Specialization is not only about different prompts. Each agent can have different doctrine, tools, connector permissions, model routing, cost limits, and role in the execution chain. That makes the workforce administratively controllable rather than simply stylistically different.

MODEL GATEWAY

Multiple AI Providers Behind One Controlled Gateway

Every LLM call flows through a common Model Gateway that handles context, routing, cost, caching and provider fallback before a provider is contacted.

The gateway first gathers relevant memory, chronicle, and playbook context, then resolves a routing policy for the specific task type. It estimates token needs, compresses context when required, checks the semantic cache, dispatches to the selected provider, records the result, and attributes cost back to the objective or work that caused the call.

Anthropic, OpenAI, Google, xAI, DeepSeek, Mistral, and Ollama can be configured at the same time. The platform does not require one globally active AI provider. Model naming and task policies determine where a request should go, and tenant-aware fallback can re-resolve to another usable provider when the intended provider has no available credential or encounters a failure. The model catalog is kept current—for example Claude Sonnet 5.5 and Haiku 5.5 and Google's Gemini 4 Argon—and announced provider price changes are applied on their effective date.

Provider keys can be stored per tenant and rotated from the application without changing the deployment .env file or restarting containers. A tenant-specific key takes precedence over a process-wide key, which makes the same installation suitable for organizations that want separate provider accounts and cost ownership.

Usage is not hidden behind the gateway

Model calls are measured for tokens and estimated cost and are attributable by agent, model, mission and other scopes. Provider-usage views distinguish real provider balance or spend data from internal estimates: the system labels confidence rather than presenting an estimate as if it came from the provider. Ollama provides a local-model path for workloads that should not use an external inference API, subject to the capability and privacy constraints of the task.

Agents Can Act Only Through Real Capabilities

A skill is the controlled unit of execution. A connector supplies reach into an external system. The Capability Engine reports whether the required capability is genuinely usable right now.

The Skills Registry contains named capabilities with input and output contracts, permissions, quality information and invocation history. When an agent decides to use a skill, the call goes through the same governed executor as work launched from a mission or external client. This gives tool use a consistent audit trail and prevents a conversational interface from becoming a bypass around policy.

The Connector Framework provides managed integrations and a configurable Connector Builder for REST-based systems. Connector operations still pass through the execution chain; connecting Gmail, Slack, Salesforce, Stripe or another system does not implicitly authorize an agent to perform every possible action on it. Credentials and action permissions remain separate concerns.

The Capability Engine then answers a practical operational question: is a named capability OPERATIONAL, CONFIGURED, DISABLED or UNAVAILABLE based on real system records? Mission planning uses that evidence rather than assuming a documented capability exists. This also helps prevent the platform from proposing that a customer buy or build something that is already configured.

MCP works through the same boundary

Outbound MCP calls are dispatched through the connector layer. The Inbound MCP Gateway issues credentials bound to an agent, which means an external MCP client cannot obtain more authority than that agent already has. Skills invoked through MCP produce the same permission checks and audit history as a skill invoked by ErezAgent itself.

CURRENT SCALE

The product ships 260+ named skills, 39 managed connectors and a 42-provider catalog of declarative API actions. The important design point is not the number: skills, connectors, chat, workers, and MCP all converge on one governed execution path.

MISSION AND DELIVERY ENGINE

From Goal or Requirement to Executable Work

ErezAgent can translate approved direction into missions, plans, dependencies, projects, and governed actions instead of stopping at a recommendation.

Mission planning is capability-aware. The system can decompose a higher-level objective into ordered work while checking what tools, skills and integrations are actually available. A mission can use AI reasoning, deterministic workflow steps, human approvals and background jobs together, and failures can be surfaced back into the attention and operating layers rather than disappearing inside an agent conversation.

For customer solution delivery, requirement documents remain traceable to their source. The platform can analyze the requirement, identify assumptions and gaps, classify whether the answer is existing ErezAgent functionality, configuration, reusable template, extension, or custom software, and generate a Solution Blueprint. The purpose is to avoid writing custom code when a governed platform capability already satisfies the requirement.

When development is required, the delivery organization can involve architecture, engineering, design, testing, security, deployment, and acceptance. Verification evidence can be attached to individual requirements so acceptance is based on what was actually delivered rather than on an unstructured statement that the project is complete.

Stage	What ErezAgent adds technically
Requirements	Source traceability, extraction, gap/assumption detection and capability fit.
Blueprint	Architecture and operating model linked to actual platform and build capabilities.
Execution	Agent and human work, skills, workflows, approvals, budgets and audit.
Verification	Testing, security evidence and requirement-level delivery evidence.
Deployment	Governed launch/deployment providers and post-deployment health checks.
Acceptance	Controlled customer acceptance and persistent delivery record.

BUSINESS TWIN

Business Twin Converts Existing Operations Into Governed Configuration

Business Twin is an onboarding and change-management capability for an existing business, not the definition of ErezAgent itself.

A business can supply files, text, or approved mission requirements. Local parsers extract content; scanned pages can use a vision model, and the analysis pipeline maps the material into typed business items such as processes, roles, record types, stages, documents, tasks, approvals, SLAs, rules, workflows, customer experiences, and integrations.

Each proposed item carries an evidence status—Verified, Inferred, Assumption, or Missing. Quote verification is deterministic rather than left to the model. Proposed items are also validated against the real platform model, so an item that cannot be represented does not quietly become an approved configuration.

The owner reviews the proposal item by item or section by section. Approved items become a version-pinned change set with dependency checks and impact analysis. Applying the change set is itself a high-risk governed execution run. Each step is read back after application, the inverse is retained where possible, and the process can be resumed or rolled back through another controlled change set.

The active Twin is later checked for drift. If configuration changes outside the Twin, ErezAgent raises the discrepancy rather than silently overwriting it. Existing business records can also be migrated between record-type versions using a previewed, resumable process that avoids firing normal workflows and notifications during the migration.

REUSABLE WITHOUT COPYING CUSTOMER DATA

Selected structures can be promoted into sanitized solution templates. The sanitizer uses an allow-list and excludes customer identifiers, contacts, documents, credentials, and other private data; installing the template still creates a draft that must pass review and approval.

OPERATIONAL APPLICATION LAYER

Configurable Business Operations Beyond a Traditional CRM

The platform includes an operational runtime for structured business processes, not only sales records or AI conversations.

Business record types can represent processes that do not fit neatly into a conventional CRM. A record type can define fields, stages, visibility, role permissions, document requirements, rules, workflow behavior, tasks and customer-facing components. This gives a venture a configurable operating model without requiring a separate custom application for every structured process.

Rules can react to record changes or time conditions. They can create tasks, send governed messages, add document requirements, invoke skills, and request signatures. Time-based rules are evaluated periodically, while the internal workflow engine uses the same classes of steps with validation, versioning, and restore. AI can therefore help interpret context while deterministic rules continue to enforce the process.

The CRM workspace brings together the customer context needed for day-to-day work: contacts, deals, projects, subscriptions, support cases, documents, appointments, activity history and next actions. Customer intelligence can add health and risk signals so the workspace answers not only “what data do we have?” but also “what needs attention now?”

Document-heavy operations are first-class

A business can define required-document checklists, accept customer uploads, classify or extract information with AI where consent permits, and route uncertain results to review. Conditional document requirements can change as the record changes. This is useful for professional services, onboarding, compliance-style workflows and any customer process where completion depends on evidence rather than a single status field.

HUMAN OPERATIONS

Staff Workspaces and Customer Portals Share the Same Business Model

ErezAgent supports human staff and external customers without exposing the owner's full operating console to either group.

Venture staff receive a separate operational identity and can be assigned role definitions that control visible and editable record types, permitted target stages, record scope, available pages and actions, and field visibility. Membership is re-read from the database on each request, so revoking access does not wait for a long-lived token to expire.

The My Work and operational workspace layers aggregate tasks, open loops, approvals, mission work, documents, and appointments around the staff member or business record. That reduces the need for an employee to navigate the entire ErezAgent platform and gives the business a way to mix human work with AI-owned work inside the same process.

Customer portals use an even narrower model. Portal components are explicitly configured, and data endpoints are allow-listed. A field must be marked customer-visible, and the relevant portal component must be enabled before the customer can see it. Record ownership is re-derived on each request rather than trusted from a link, which is important for preventing cross-customer access.

The portal can expose stage progress, document requests and uploads, forms, messages, signatures, appointments and selected actions. This allows a business to give customers useful self-service access without giving them access to internal staff workspaces or the broader AI operating environment.

ONE MODEL, THREE EXPERIENCES

Owner/operator users manage the whole organization, venture staff operate focused processes, and customers interact with explicitly exposed portal components. The underlying business records remain the same, but identity and authorization determine which experience is rendered.

CUSTOMER OPERATIONS

Support, Customer Success and Appointments Are Operational Systems

Customer-facing work is handled through state, evidence, and governed actions rather than through free-form AI replies alone.

Support Department 2.0 uses a real incident lifecycle with constrained states and status history. Resolution requires verification evidence; if a proposed solution has not been verified, the case remains a proposed resolution instead of being marked solved. Diagnosis can incorporate real connector health and recent platform errors, so AI drafting is grounded in operational evidence.

Remediation actions such as connector reconnect, diagnostic rerun, or runbook execution go through the same governance machinery as other consequential actions. Escalations create real records and can hand work to a human when no appropriate AI specialist exists. Verified resolutions can be promoted into the knowledge base only after human approval.

Customer success builds on the customer operating picture: health, churn risk, renewals, success tasks, interaction history, and next actions. The appointment engine adds real Google Calendar/Meet integration, public booking, preparation checks, meeting briefs and structured post-meeting extraction. Follow-up communication remains human-approved rather than being silently sent by the AI.

OPERATIONAL PRINCIPLE

The customer layer uses AI to understand, summarize, and recommend, while lifecycle state, access, SLAs, reminders, and approvals remain structured system behavior.

BUILD AND LAUNCH

Software Delivery Is Connected to the Business Operating Model

When a mission genuinely requires software, ErezAgent can coordinate development and deployment as part of the same governed organization that defined the business objective.

The engineering layer can maintain project work, engineering roles, sprint activity, technical debt, and repository intelligence. Generated or customer-delivered software can move through build, test, fix, security, and verification loops instead of being treated as one-time generated code. Security standards and project certification can be included before release.

The Launch Orchestrator coordinates the steps around turning software into a live venture: brand, scaffold, deployment, DNS, payments, email, analytics, revenue, and outcome tracking. Deployment-provider abstraction allows the user to choose where a venture should run rather than letting the system silently pick an infrastructure target. The current product documentation includes local deployment plus Vercel, Railway, Fly.io, and Hetzner paths.

Rollback is treated as an execution concern rather than a hopeful manual recovery step. Orchestrations can use a compensation stack so completed steps are reversed in LIFO order when a later step fails.

Compensation failures are surfaced rather than hidden, which is important when deployment work spans several external systems.

Security is part of delivery

CI and security assurance can scan source, dependencies, secrets, and container configuration. Critical CVEs or secret leaks can block publish or deploy. AI can propose remediation, but a separate verification step reruns the relevant scanner before a finding is considered resolved.

Go-to-Market Uses Real Pipeline and Economics Data

Marketing and sales capabilities are connected to the same ventures, customers, deals, experiments, and financial records used elsewhere in the platform.

A GTM Motion represents the durable way a venture sells—for example, founder-led, outbound, inbound, enterprise direct, partner, MSP/channel, reseller, referral, or managed service. Offers, segments, and growth experiments can be linked to those motions so the system can compare actual performance instead of treating every campaign as an isolated activity.

The GTM intelligence layer aggregates active motions, offers, segments, funnel data, and channel economics from existing business records. Deterministic detectors can identify channel/offer mismatch, customer-segment behavior shifts, expansion candidates, and missing evidence without spending tokens on an LLM call. The resulting evidence can be fed into Executive Board deliberation.

Sales and revenue workflows can include prospect research, scoring, outreach drafts, pipeline management, deals, renewals, and campaign performance. Consequential actions continue through the normal approval gates. This means “commercial department” does not imply an autonomous bot that can spend, promise, publish or contact customers without policy.

WHY THIS MATTERS TECHNICALLY

The same data model links market hypotheses, experiments, deals, invoices and outcomes. That allows ErezAgent to evaluate whether commercial activity created measurable value rather than reporting activity volume as success.

Commercial Operations Are Built on an Explicit Usage and Entitlement Chain

The billing foundation supports direct customers, ventures and provider/MSP operating models without the self-hosted product reporting anything by default.

The usage ledger is append-only and records AI usage and cost, executed actions, agent runs, missions, capacity snapshots, value-share outcomes, and venture-defined units with venture and project attribution where known. Statements are derived from that ledger and become immutable when finalized. Invoice drafts can then include plan, usage, discount, tax, adjustment, and credit-note lines using prices configured by the operator.

Payments can be processor-confirmed, owner-attested, or imported from a bank CSV, with support for partial payments and reversals. Billing documents can be rendered as PDF or DOCX in English, Hebrew, and Arabic and sent through a governed email action. The system explicitly calls these usage statements and invoice drafts—not jurisdiction-specific tax invoices.

Entitlements are resolved through one decision path: platform-owner instance, verified installed license, active subscription, active trial, or no entitlement. Licenses carry feature areas plus limits for ventures, active projects, users, client workspaces, and staff seats, with optional add-ons such as white label, OEM, multi-tenancy, local LLM, metering export, and API access. A customer is not tied to the plan chosen at install: the plan and limits of a license come from the license file, and a subscription can be changed, upgraded or downgraded later.

Provider/MSP operation

The platform owner can maintain downstream customer contracts, invoices, payments, renewals and license updates. Self-hosted installations do not report usage back to the vendor automatically; usage reporting is exported by the customer and imported on the vendor side. License-signing keys can be rotated through a certificate chain so upgraded customers can continue to trust newly issued licenses without manual key replacement.

Billing/commerce surface	Implemented behavior
Billing method	Metering or configured gateway; owner-governed changes.
Gateways	Stripe, PayPal and Rapyd for plan flows; Pelecard for supported deal-payment scenarios.
External contract	Signed webhooks, scoped read/ingest tokens and JSON/CSV export.
Documents	Usage statements and invoice drafts rendered to PDF/DOCX; not legal tax invoices.
Provider side	Customer contracts, renewals, license updates, payment tracking and usage-file import.

PROVIDER CONTROL PLANE

Providers See Status, Customers Keep Control

An optional, consent-based link lets a provider see the operational state of the installations it enrolled, without any remote control of customer servers.

The provider creates an installation record for a customer and hands the customer owner a one-time enrolment string. In Settings, the owner pastes it, sees the provider’s address and the exact lists of what is and is not shared, and agrees. Nothing is sent before that, and the owner can pause, resume or disconnect at any time; a disconnected or restored installation does not keep reporting.

The installation then sends a signed status report about every fifteen minutes: ErezAgent version and migration head, workspace count, core-service status, a count-level security summary, AI and voice provider status, connector counts, readiness state, resource flags, and licence and subscription state. It never sends prompts, model responses, business data, documents, e-mail, CRM data, names, credentials or keys, and the page shows the exact last report. If a later release widens the list of shared items, reporting pauses until the owner confirms again.

On the provider side, a Fleet page lists every enrolled installation with filters for offline, unhealthy, update available, unsupported version, security attention, AI provider issue, licence expiring, subscription attention and support required. The provider can send plain-text notices, which the customer answers (I will update

myself, remind me later, request assistance, request professional services); the last two open a support case. Service packages are informational, and a deterministic service report states only stored facts.

Provider staff roles (operations, support, security, billing, auditor) use a separate session scope, are limited to the customers the owner assigns, and are re-checked on every request.

Aspect	Implemented behavior
Consent	The owner enrolls, sees the address and the lists, and agrees; revocable at any time.
Report	Signed (Ed25519), status only, about every fifteen minutes.
Provider view	Fleet filters, notices, packages, support cases, service report.
Access	Separate staff scope; role and customer scope re-read on every request.
Not included	Remote update, backup, shell or log collection.

BOUNDARY

The channel reports status and returns information. The provider cannot update, back up, run commands on, or collect logs from a customer server; the customer applies updates locally.

SECURITY AND PRIVACY

Security Controls Are Embedded in the Runtime, Not Added as a Separate Dashboard

The platform’s security model combines database isolation, role enforcement, encrypted credentials, governed side effects, secure-delivery gates and privacy controls around AI analysis.

Tenant-scoped data is protected by PostgreSQL Row-Level Security with FORCE RLS on tenant tables and request-scoped tenant context. The application also applies explicit service-layer scoping in sensitive queries. The architecture deliberately treats tenant isolation as a database concern rather than relying solely on developers remembering to add a tenant filter to every query. The application’s database role cannot edit or delete audit-log rows.

RBAC separates owner, operator and observer behavior. Observers are centrally enforced as read-only. High-impact operations such as destructive deletes, security containment, secret rotation, licensing changes and many financial changes are owner-only. Venture staff use a distinct identity scope and are accepted only by the specific operational routes designed for staff work. Provider staff on a provider installation use another distinct scope limited to the Fleet page and the customers assigned to them. A workspace uses the installation owner’s provider accounts only when it is the home workspace or the only one.

Provider keys and connector credentials are encrypted at rest. Passwords use Argon2id, sessions are invalidated on logout or password change, and password reset links are single-use. CI includes linting, tests, dependency scanning and secret scanning; critical security failures can block release activities rather than appearing only as advisory findings.

AI analysis has an explicit privacy mode

AI-analysis consent is owner-controlled. Redaction covers categories such as Israeli ID numbers, IBANs, payment-card numbers, phone numbers, email addresses, and secrets before relevant document content leaves the platform. In local-only mode, the operation must use a private local model or fail closed; it does not silently fall back to a cloud model. The documentation is also explicit about what masking does not reliably detect, such as all national ID formats, personal names, and free-text addresses.

CONTROLLED AI, NOT UNRESTRICTED AUTONOMY

Security is enforced at several choke points: database isolation, identity and role checks, capability grants, action risk, approval policy, spending policy, connector execution, CI/release gates and privacy routing. No single prompt is expected to provide all of those controls.

DEPLOYMENT AND OPERATIONS

Self-Hosted by Design, With a Real Installer and Operational Readiness Model

ErezAgent is packaged as a multi-container application rather than as a thin hosted chat client, and the installation lifecycle is treated as part of the product.

The runtime uses a React/TypeScript frontend, FastAPI API, PostgreSQL 16 with pgvector, Redis 7 and ARQ background workers, with additional service containers for connectors, security, sandbox execution, document rendering and observability depending on the deployment. Production deployment pulls prebuilt images, while the source repository can also build services locally.

Windows, Linux, and macOS installation is supported through a standalone web installer, with shell and PowerShell paths also available. The installer checks Docker, collects identity and provider configuration, generates cryptographic secrets in-browser, assembles the environment file, launches the containers, runs health checks and supports trial or license activation. Installer downloads are published with a checksum file.

Upgrade behavior includes environment-key parity checks, database migration handling, image verification and rollback support. Rolling back to an earlier version can also roll the database schema back, first showing what will be removed and creating a database backup. The expected migration head is tracked so System Health can identify a stale or incompletely migrated installation. Monitoring services read container information through a filtered, read-only Docker socket proxy; monitoring images are pinned by digest and containers report their own health.

Readiness is evidence-based

The Customer Readiness Check combines system health, provider connectivity, security assurance, tenant-isolation diagnostics, capability evidence, and backup state into a five-state report: VERIFIED, CONFIGURED_NOT_VERIFIED, NOT_CONFIGURED, FAILED, or NOT_APPLICABLE. The check deliberately distinguishes “configured” from “proven to work” and exercises the approval path using a no-side-effect probe.

Resource profile	CPU	RAM	Disk
Minimum	4 cores	8 GB	40 GB SSD
Recommended	8 cores	16 GB	100 GB SSD
Large	16 cores	32 GB	250 GB SSD

OBSERVABILITY AND BACKGROUND OPERATION

The Platform Continues Working Between User Conversations

ErezAgent uses background workers, real-time events, and health systems so operations do not depend on an open chat session.

ARQ workers run domain jobs for governance, orchestration, mission execution, venture health, connectors, social listening, finance, growth, security, backup, knowledge, support, appointments, Business Twin, time-based rules, and maintenance. This is where recurring operational behavior lives; a scheduled rule or health check is not simulated by asking an LLM to remember to do something later.

The frontend maintains one Server-Sent Events connection that routes mission, agent, approval, and other events into domain stores. This lets the command center reflect execution state without polling every subsystem independently. The same event stream drives the visible AI presence state and operational updates in the interface.

Prometheus, Grafana, and Loki provide platform observability, while the application itself exposes System Health, benchmarks, support bundles, update planning, recommendations, backups, and platform diagnostics. Data-retention cleanup is fail-closed when the required recent backup is missing, which prevents a maintenance job from destroying data simply because its schedule fired. Log-like tables (model calls, runs, error and audit records) follow retention windows, and the first run after an upgrade only reports what it would remove.

OPERATIONAL CONTINUITY

ErezAgent is designed to keep executing governed jobs, monitoring conditions, evaluating time-based rules, and surfacing attention signals when the user is not actively chatting with it.

What the Technical Claims Mean — and What They Do Not Mean

A useful technical guide should make the edges of the product clear rather than treating every capability as unlimited automation.

“Self-hosted” means the ErezAgent application and its primary data stores can run in customer-controlled infrastructure. It does not mean every external capability is local; a connected cloud AI provider, email service, payment gateway or SaaS connector still sends the data required for that integration. Local-only AI mode is available for supported private workloads but intentionally fails rather than falling back to cloud when privacy policy requires locality.

“Build software” means the platform can coordinate requirement analysis, blueprinting, engineering work, testing, security verification, deployment and acceptance through its delivery system. It is not a guarantee that every arbitrary software request can be completed without human engineering judgment, external credentials, infrastructure access or remediation work.

“Billing” means usage metering, statement generation, invoice drafts, payment evidence, subscriptions, entitlements and provider/customer operational flows. ErezAgent does not claim that its generated invoice documents are legal tax invoices in every jurisdiction. The operator remains responsible for local invoicing and tax requirements.

“Governed autonomy” means the system can execute low-risk work automatically when policy permits while forcing higher-risk actions through approval or blocking them entirely. It does not mean every action requires a click, and it does not mean agents can bypass a control because the user asked persuasively.

“Business Twin” means an evidence-backed model and governed configuration mechanism for the parts of a business that ErezAgent can represent. It does not imply that every undocumented behavior, legal obligation or informal human practice can be inferred perfectly from uploaded material.

“Provider connection” means optional, owner-consented status reporting and notices. It is not remote management: it does not update, back up or administer a customer installation, and it reports nothing until the owner connects it.

A Focused View of the Technical Surface

The table below summarizes the major product domains described in this guide. It is a product capability map, not an endpoint inventory.

Domain	Core capability	How it is controlled
Strategy	Goal shaping, objectives, Executive Board, measurable value	Human confirmation, evidence, Board governance
AI workforce	46 specialized agents, persona, delegation, human/AI ownership	Per-tenant overlays, skills, policy, budgets
Model gateway	Multi-provider routing, fallback, caching, cost attribution, local models	Task policies, tenant keys, privacy mode
Execution	Skills, actions, workflows, missions, background jobs	Autonomy tiers, approvals, spend limits, audit
Integrations	Managed connectors, REST Connector Builder, inbound/outbound MCP	Credential isolation, grants, governed side effects
Business Twin	Evidence-backed operating model, change sets, migration, drift	Owner review, version pinning, high-risk apply
Operations	Configurable records, CRM, documents, rules, tasks, appointments	RBAC, field visibility, deterministic workflow
Staff & portal	Venture staff runtime and customer self-service	Separate identities, allow-listed components, record ownership
Customer ops	Support, success, health, SLA, escalations, appointments	Lifecycle state, verification, human escalation
Delivery	Requirements, blueprint, build/test/security/deploy/acceptance	Governed project flow and verification evidence
GTM	Motions, offers, experiments, pipeline, channel economics	Real business data, approval-gated external actions
Billing	Usage ledger, statements, invoice drafts, payments, entitlements	Owner controls, signed webhooks, scoped tokens
Provider fleet	Consent-based status reports, Fleet page, notices, support cases, service report, staff roles	Owner consent, signed reports, role and customer scope, no remote control
MSP/provider	Downstream licenses, renewals, customer billing, usage import	Signed licensing; usage exported by the customer; optional consent-based status reporting
Security	RLS, RBAC, credential encryption, scanning, containment, privacy routing	Code-level gates and release controls
Platform ops	Installer, health, readiness, backup, upgrade, rollback, observability	Evidence checks, migration head, backup-first rollback

THE COMMON ARCHITECTURE

Across these domains, ErezAgent repeatedly uses the same design pattern: AI interprets context and proposes action; deterministic services hold state; execution gates decide authority; evidence and audit record what happened; measured outcomes feed the next decision.

What a Technical Buyer Should Evaluate in a Pilot

The most useful proof of ErezAgent is not a feature tour. It is whether the complete chain works on a real business process under the buyer's own security and operating constraints.

A meaningful pilot should begin with one measurable outcome and one real operational process. Configure the required AI provider, connector and staff roles, then use Start or Business Twin to create the initial context. The pilot should deliberately include at least one action that can run automatically and one action that must require approval so the governance model is exercised rather than merely observed in screenshots.

The buyer should then verify the complete evidence chain: the objective is measurable; the mission or process is created from approved context; the required capability is genuinely operational; an agent or workflow performs work through the governed executor; the result appears in the business record or external system; cost and audit data are captured; and the outcome feeds the operating view.

For an MSP or enterprise evaluation, the same pilot should additionally test tenant isolation, a venture-staff identity, customer portal exposure, backup/readiness behavior, license entitlement, an upgrade or rollback path, and the external billing contract if it will be used. If a provider connection is planned, the pilot should also test consent, a status report, a notice and its answer, and disconnecting. This tests the product as an operating platform rather than as a demonstration chatbot.

RECOMMENDED SUCCESS QUESTION

At the end of the pilot, ask: "Can this system take a real business objective, execute useful work across our systems, keep authority under our control, show us what happened and what it cost, and continue operating after the chat window closes?" That is the technical proposition ErezAgent is built to prove.

EREZAGENT

Governed AI execution from intent to measurable value.