FOR IT & INFRASTRUCTURE TEAMS

Corvexa — Technical Specification & System Requirements

A reference for what it actually takes to host and run Corvexa: runtime and server requirements, network and security architecture, data storage, and the web interfaces operators, administrators, and visitors use — so your team can plan a deployment before a single license is signed.

Deployment Model

Corvexa is a self-hosted application that runs on infrastructure your organization controls — it is not a multi-tenant SaaS product. It's deployed on a server or virtual machine on the same network (or a routed/VPN-connected network) as the video, access control, intrusion, and life-safety subsystems it integrates with, since those systems typically live on a private site network rather than the public internet.

On-premises — the primary, expected model: a server or VM inside your own facility network, alongside the subsystems it connects to.

Customer-controlled cloud — a VM in AWS, Azure, or any other cloud is fully supported, provided network connectivity back to on-site subsystems is solved (site-to-site VPN, Direct Connect / ExpressRoute, etc.).

Server & Runtime Requirements

COMPONENT	REQUIREMENT
Runtime	`Node.js ≥ 18.0.0 (Node 24 LTS recommended)`
Operating system	Any OS Node.js runs on (Linux, Windows, macOS). Linux is the typical production target.
Application dependencies	`express, qrcode, pg`
Database server	**Optional.** Default: none — file-backed persistence under one data directory. Or **PostgreSQL 14+** (tested on 16); see Data & Storage.
Web server (application)	Built-in Node/Express HTTP server. Speaks plain HTTP only — TLS is terminated externally (see Network & Security).
Reverse proxy	Required for production. nginx and Caddy reference configurations provided.
Outbound connectivity	To each integrated subsystem's own network endpoint (VMS, access control, intrusion, BAS) — scoped per connector, not a blanket internet requirement.

Dependency footprint is deliberately minimal by design — everything beyond HTTP routing (express), QR codes (qrcode) and the optional PostgreSQL driver (pg) uses Node's standard library — fewer packages to patch or audit.

Suggested Hardware Starting Point

Not a vendor-certified minimum. No fixed CPU/RAM/disk requirement is published — Corvexa is a lightweight, single-process application, and the right size depends on device count, event volume, and how many operator/admin sessions stay connected simultaneously. The figures below are a reasonable starting point for a planning conversation; our Site Assessment (see back page) right-sizes this to your actual environment.

Measured, not just estimated. Under a burst of simultaneous incidents across every connector type, the application process itself used roughly 50–60 MB of memory (on file storage or PostgreSQL) and well under one second of CPU time. The vCPU/RAM figures below are sized for headroom — the reverse proxy, OS, and multiple concurrent operator sessions — not because the application itself is resource-intensive.

DEPLOYMENT SIZE	ILLUSTRATIVE STARTING POINT	TYPICAL FIT
Single site	2 vCPU · 4 GB RAM · 20 GB disk	One facility, a few dozen devices, one operator console at a time
Multi-site / campus	4 vCPU · 8 GB RAM · 50+ GB disk	Several buildings or sites, higher event volume, multiple concurrent consoles

Disk usage grows with event-log retention and uploaded floor-plan images (see Data & Storage) — size for your retention policy, not just today's footprint. Old records are archived, not deleted; see Data & Storage.

Network & Security Architecture

ASPECT	DETAIL
Default listen port	`4000 (HTTP)`
TLS	Not terminated by the application. Terminate HTTPS at a reverse proxy (nginx reference config provided) and proxy plain HTTP to the app.
Real-time updates	Server-Sent Events (SSE) on /api/stream — no WebSockets are used anywhere, which simplifies firewall and proxy rules. **Your reverse proxy must disable response buffering and allow a long read timeout on this route**, or real-time push breaks silently. A browser cannot send an authorization header on this stream, so the page first asks for a **ticket** (with its header) and opens the stream with that: it works once, for the stream only, expires in 30 seconds and is tied to the address that asked for it. No session token appears in any URL or proxy log; every other route, including floor-plan images and printable reports, requires the Authorization header.
Request body limit	Most requests are limited to 64 KB. The few larger ones have their own limits: a floor-plan image upload 15 MB (a base64-encoded image of up to ~11 MB), a CSV import 2 MB, a kiosk badge photo 4 MB. The reference proxy configurations cap any body at 16 MB.
Authentication	**Local accounts** (recommended): 12+ character passwords (salted scrypt), lockout per account and source address after repeated failures (so a stranger guessing cannot lock the owner out), an authenticator-app code required of **every role** by default, single-use recovery codes and a one-time enrollment code for a new account's first authenticator, server-side sessions that sign-out, an admin or a role change end at once, all audited. **Stand-in sign-in** (development and demos only): account id, no password; protect it another way in production. **Single sign-on (AD, SAML, OpenID Connect) is not built yet**; OpenID Connect is the planned first step.
Authorization	Three-tier role model — operator < supervisor < admin — enforced on every route.
Kiosk access	Each lobby check-in tablet is registered by an administrator and presents its own secret; in production the public kiosk endpoints refuse a device that is not registered (and can also be limited by source address).
Release integrity	Each release package comes with a checksum, a signature (Ed25519) that you verify against E2's public key sent by a separate route, and a software bill of materials (CycloneDX) listing every dependency with its version and hash.
Admin workstation restriction	Optional: restrict the admin role to a single allow-listed source IP.
Console IP allowlist	Optional: restrict the entire authenticated console/API to a set of allow-listed source IPs (the public visitor-kiosk endpoints are exempt by design).
Production safety check	The server refuses to start in production if default signing secrets or a development auth bypass are still in place, or if it would run the password-less stand-in sign-in, run without the audit-trail signing key, take backups without an encryption key, or run a simulated connector or notification channel — each of those can be acknowledged on purpose for a lab or demo server — so a misconfiguration can't accidentally reach production silently.

Web Interfaces & Browser Requirements

Corvexa is delivered entirely as web pages — there is no desktop or mobile app to install. Every interface runs in a standard, current browser.

INTERFACE	WHO USES IT	BROWSER / DEVICE REQUIREMENT
Operator Console	Operators, supervisors, admins	Any current evergreen browser (Chrome, Edge, Firefox, Safari) with Server-Sent Events support — standard in all of them.
Visitor Kiosk	Front-desk / lobby tablet	Same, **plus** camera access (getUserMedia) when the organization requires a badge photo — this also requires the page be served over HTTPS, which the standard reverse-proxy setup already provides. Visitor check-out is an organization setting: by an operator, by the visitor scanning the badge's barcode at the kiosk (any keyboard-wedge USB or Bluetooth barcode scanner), or automatically after a set time.
Guard Tour App	Patrol / mobile guard staff	Any current mobile or desktop browser — a lightweight page, no camera or location APIs required.
On-Call App	On-call / after-hours staff	Any current mobile or desktop browser with Server-Sent Events support.

No native app install · No plugins · No WebSockets to allowlist

Data & Storage

Corvexa stores its data one of two ways, chosen at install time. By default everything lives under a single data directory with no database to install. Optionally the same data lives in PostgreSQL instead — the choice to make for larger event volumes, failover, and database-native backups.

DATA	DEFAULT: FILES	OPTIONAL: POSTGRESQL
Event log	One append-only NDJSON file per day; older days compressed	Table partitioned by month
Audit trail	Append-only NDJSON, hash-chained and signed with a key kept off the server	Append-only table, signed the same way; the database itself refuses edits and deletes
Incidents, visits, guard-tour runs	Journal files	Tables, stored exactly as written so integrity seals still verify
SOPs, users, org settings, calendar	JSON files	Rows in a configuration table
Floor plan images	PNG / JPG / SVG files under the data directory, one per building, up to ~11 MB each (either option)	
License key	Signed key file, installed once per deployment (either option)	

Retention archives; it never deletes. Records past a configurable window (90 days by default; incidents, visits and tour runs, and the event log, each have their own) move out of the working set into an archive — compressed day files, or archived rows — and stay in every export and range query. A legal hold pauses all archiving, and the audit trail is never archived. Administrators can export any date range as JSON or CSV for audits and records requests, and can erase a visitor's personal details on request (the erasure is audited without the person's name, and does not reach backups made before it, which are replaced as newer ones run).

Tamper-evident. Every audit record carries a hash of the one before it and a signature only the holder of the audit key can make, and each resolved incident is sealed with a fingerprint recorded in the audit trail. Signed checkpoints of the trail are also written to another disk or a syslog server, so deleting the newest records is caught as well. A built-in check reports exactly where a record was altered, removed or reordered.

Plan disk for your event volume. As a planning input, not a guarantee: raw events run roughly 500 bytes each on file storage and roughly 1 KB each in PostgreSQL including indexes, and audit entries roughly 200 bytes each on file storage, in measured testing — multiply by your site's expected daily event volume (a function of real device count and activity, not the software), then by your retention windows, to project actual disk growth.

Availability, Backup & Recovery

CAPABILITY	DETAIL
Scheduled backups	Set BACKUP_DIR to a different disk, share or write-once location. Every install gets incremental snapshots of its data directory with a checksum manifest; a PostgreSQL install also gets pg_dump backups (the PostgreSQL client tools must be on the host). Backups are encrypted (AES-256-GCM) with a key held away from the backup location. Each backup is verified as it is written, and the newest N are kept (a legal hold suspends removal).
Restore drill	A built-in drill restores the newest database backup into a scratch database, re-runs the integrity checks on the restored data, and removes it. Measured on a 3.9 GB test database (4 million events): backup in under a minute, drill in about two and a half.
Moving to PostgreSQL	A scripted cutover copies everything across and verifies it — record counts, the audit chain, every incident seal — before switching; the original files are left read-only. A scripted rollback brings them back up to date if you change your mind. Rehearsed end to end on a copy of a real data directory.
Failover (optional)	On PostgreSQL, two or more application servers can share one database as active/passive failover: one leads, the others wait and take over within seconds (about 1.6 seconds measured in a two-process test), restoring open incidents and tours. A node that loses its database connection, or was frozen past its lease, refuses to write and steps down. Put a load balancer that health-checks /api/health in front. It is failover, not scale-out; a failover drops in-progress kiosk sessions.
Database availability	A highly available PostgreSQL itself (a replica, or a managed service) comes from your database platform, not from Corvexa.
Not covered by backups	The .env secrets file, which holds the audit and backup keys — keep it safe separately: without the backup key the backups cannot be read. A runbook covers provisioning, database roles, TLS, cutover, rollback, backups and monitoring.

Key Configuration Variables

VARIABLE	PURPOSE	REQUIRED?
PORT	HTTP listen port	Optional — default 4000
NODE_ENV	Set to production to enable startup safety checks	Optional — default development
DATA_DIR	Root directory for persisted data (all of it on the default file storage; floor plans and the license key either way)	Optional — default ./data
STORAGE_BACKEND	files (default), postgres, or shadow (files stay in charge while every write is also copied to PostgreSQL, to prove it out first)	Optional — default files
DATABASE_URL	PostgreSQL connection string (use an application role that cannot alter audit records)	**Required for postgres**
BACKUP_DIR	Where scheduled backups go — a different disk, share or write-once location	Optional — backups are off until set
HA_MODE	Lets several servers share one PostgreSQL database as active/passive failover	Optional
JWT_SECRET	Signs session tokens	**Required in production**
AUDIT_HMAC_KEY	Signs the audit trail; kept off the server's data	**Required in production**
BACKUP_ENCRYPTION_KEY	Encrypts every backup; kept off the backup location	**Required when BACKUP_DIR is set**
AUTH_MODE	local: passwords + authenticator app (the default is a development stand-in)	**Set to local in production**
SESSION_TTL_HOURS	Longest a sign-in lasts	Optional — default 12
TRUST_PROXY	Enable when running behind a reverse proxy, so client IPs resolve correctly	Optional
KIOSK_ALLOWED_IPS	Comma-separated allowlist for the public visitor-kiosk endpoints	Optional
CONSOLE_ALLOWED_IPS	Comma-separated console/API IP allowlist	Optional
ADMIN_WORKSTATION_IP	Single IP the admin role is restricted to	Optional
*_MODE	Per-connector mock/live switch (video, access control, intrusion, BAS, gunshot detection, weapon detection) — lets a connector run simulated before it's pointed at a live system	Optional — default mock

Each live connector (video management, access control, intrusion receiver, BAS email ingestion) takes its own endpoint and credential configuration once you're ready to point it at a real subsystem — scoped during systems integration, not listed exhaustively here.

Licensing

Corvexa is licensed per site. Corvexa Core is a site license plus an annual fee per device (a camera or an access-controlled door), and optional modules — burglar and fire alarm, building automation, gunshot detection, weapon detection, guard tours, visitor management — are flat per-site fees; see the Platform Pricing & Modules sheet. The signed license key is time-based, tied to your organization name and to the one installation it was issued for, and enables exactly the modules you license. It is **not** metered by concurrent user seats, so adding operators or consoles does not change the license; capacity planning for them is a hardware and network conversation (see Suggested Hardware Starting Point).

What This Means for Your Rollout

- No database server needed to start; PostgreSQL is there when you want failover and database-native backups, with a scripted, verified way to move over.
- Backups you can prove: every backup is verified, and a restore drill shows it can actually be restored.
- A single reverse proxy (config provided) handles TLS and the one SSE-specific setting that matters.
- New connectors run in simulated mode first — your team validates correlation logic and SOPs before a single credential touches a live system.
- Everything your operators, admins, and visitors use is a web page — no fleet of native apps to package, sign, or push updates to.

Ready to scope your environment?

Our Site Assessment reviews your subsystems, network topology, and expected event volume, and right-sizes the deployment before anything is installed.

info@e2securityconsulting.com
(754) 732-8290

© 2026 E2 Security Software & Consulting. All rights reserved.