EDCS Deployment and Operations Guide

Enterprise Directory and Configuration Store

November 2026 · Describes the product as of November 12, 2026. EDCS has no numbered release yet; every installation we deliver is identified by its build, which we give you with it.

1. Architecture

EDCS is a set of small services that run as containers on a private network behind your reverse proxy, which terminates Transport Layer Security (TLS). The directory, vault and configuration APIs accept only requests that carry a valid token with the right scope.

Service	What it does
Sign-on service	The hosted sign-in pages and the OAuth 2.0 / OpenID Connect token service (browser sign-in uses Proof Key for Code Exchange, PKCE): passwords, second factors, security keys and passkeys, Common Access Card (CAC) and Personal Identity Verification (PIV) smart cards, sign-in through partner organizations, the Security Assertion Markup Language (SAML) 2.0 identity provider, sessions and sign-out, signing keys.
Directory service	A web application programming interface (API) over the Lightweight Directory Access Protocol (LDAP) directory of people and groups; the System for Cross-domain Identity Management (SCIM) 2.0 provisioning endpoint and outbound connectors; governance (entitlements, requests, attestation, separation of duties) and the policy decision point applications ask for access decisions.
Vault service	Encrypted storage of application secrets, with versions.
Configuration service	Application settings and feature flags; settings that point at a vault secret are resolved when an authorized application reads them.
Administration console	The web console for administrators and delegated administrators.
Migration runner	A one-time job that applies database schema changes for the shared registry and for every organization before the services start.

Supporting components you provide or that run alongside EDCS:

- **A standards-based LDAP directory server** that holds people, groups and passwords. Two widely used open-source directory servers are supported.

- **A relational database server.** Each service keeps its data in its own database, and each organization (tenant) gets its own set; a control-plane registry maps web addresses and organization codes to organizations.
- **A reverse proxy or load balancer** that terminates TLS. For smart-card sign-in it also requests the client certificate, on a host name dedicated to that purpose.
- Optional: a hardware security module (HSM) for signing keys, a security information and event management (SIEM) system for audit export, and a mail relay for sign-in codes and links.

2. Single-node deployment

The simplest installation runs every service on one host, each in its own container on a private network, with the directory server and database on the same host or on servers you already run. The shipped deployment files run each service with a read-only root filesystem, every Linux capability dropped except DAC_OVERRIDE (which the sign-on and vault services add back), privilege escalation blocked, and a memory limit. Each image carries its own health check. Not finished: the sign-on service and the vault still run as the root user in the shipped deployment files, with that one capability added back, because they read host files owned by another user; four more services run as root by default until a host passes a documented check. The *EDCS Security and Standards Guide* lists this under hardening.

A single node is the right starting point for an evaluation and for small organizations. Its limit is the obvious one: a host restart is an outage of sign-in for every application that depends on it.

3. Multi-instance high availability (within one site)

Every service can run as several instances behind a load balancer. The state that must be shared lives in the database, so any instance can continue any session:

- Signing keys, refresh-token families, authorization codes, pending SAML requests, one-time codes, sign-in links and revocations are already in the sign-on service's database.
- The key ring that protects session cookies and stored authenticator secrets can be stored in the database, sealed per service, so replacing an instance does not sign anyone out.
- Lockout counters and rate-limit windows are shared, and decided on the database's clock, so an attacker gets the limit once, not once per instance. If the database cannot be reached, instances fall back to counting on their own (never to no limit) and log a warning.
- Background jobs that must run once (audit checkpoints, SIEM export, the governance sweep) hold a lease; a dead holder is replaced automatically.
- The administration console keeps a live connection to one instance, so the load balancer needs sticky sessions for it. If that instance dies, its users reconnect to another and stay signed in. The other services need no stickiness.

What was tested. A failover test starts two instances of every service behind a proxy, proves they share state (a sign-in started on one instance and finished on the other, a lockout earned across both, one rate-limit window counted across both), then runs sign-ins, refreshes and API calls from several clients while one

instance of every service is killed part-way through. The run fails on any failed operation. It is a functional failover test, not a load test.

What is yours. High availability of the directory server and the database is part of your infrastructure. Replication, load balancing and failover **between sites or data centers** are not part of EDCS today.

---

4. Government-cloud deployment template

An infrastructure-as-code template describes EDCS in a commercial cloud provider's US government cloud regions (the runbook names the provider and services):

- A virtual network across two availability zones. Public subnets hold only the load balancers and outbound gateways; services run in private subnets; the database and file storage sit in data subnets with no route to the internet. Security groups allow only the connections the services make.
- Two instances of every service (one for the directory server) with read-only root filesystems, health checks and automatic rollback of a failed deployment.
- A managed database deployed across two availability zones, encrypted with a customer-managed key, reachable only over TLS, with automated backups (14 days by default) and deletion protection.
- A web application firewall with managed rule groups and a per-address rate limit, and TLS 1.2 or later on the load balancers.
- A **separate entry point for smart cards**: a load balancer that runs mutual TLS against a trust store of the certificate authorities you supply, forwarding only to a small proxy beside the sign-on service, which adds the secret the sign-on service requires before it believes a forwarded certificate. A forged certificate header sent to the main entry point is ignored, logged and raises an alarm.
- One secret per value in the managed secrets store (values are written by an operator and never appear in the template or its state), three customer-managed encryption keys with rotation, log retention of at least 365 days, and alarms for server errors, services with no healthy instance, processor and memory use, database storage and memory, a spike in blocked requests, the forged certificate header and any fatal error.

Status. The template passes validation, offline tests and static security scanners. **It has not been applied to a cloud account**, so the first real plan may surface details the offline checks cannot see. Two trade-offs are stated in the template's runbook: the directory server runs as one instance on network storage, so replacing it is a short directory outage; and the vault's key-encryption key is delivered from the secrets store rather than wrapped by the cloud's key management service. A government cloud region does not by itself give any authorization (see "What is not included").

5. Backup and key recovery

What	How to protect it	What happens without it
Databases (each service, each organization, and the registry)	Back up each database on a schedule; the managed-database template does this automatically. The vault database contains only encrypted data and is safe to back up as it is.	Data since the last backup is lost.
The vault's key-encryption key passphrase	Keep it in a separate, offline place, apart from the database backups.	**Secrets cannot be recovered.** Restoring the vault needs both the database backup and this passphrase.
Directory data	Export to LDAP Data Interchange Format (LDIF) online, or snapshot the volume offline. Restore into a running, base-seeded directory.	People, groups and passwords are lost.
Session key ring and its master secret	Back up the master secret that seals the key ring.	Everyone is signed out, and stored authenticator-app enrollments can no longer be read, so people enroll again.
HSM signing keys	Use the HSM vendor's own backup or replication; EDCS cannot export these keys, by design. Several sign-on instances must share one token or a replicating HSM cluster.	Rotate to new keys: tokens and SAML assertions signed with the old keys stop verifying, and partners refresh metadata.
Audit checkpoint keys	Keep them off the database host, and keep retired keys for verification only.	Older checkpoints can no longer be verified.

Run a restore on a test installation before relying on any backup.

6. Upgrade and migration approach

- **Schema changes run first.** The migration runner applies the registry's and every organization's database changes before the services start. Enabling multiple organizations on an existing installation is non-destructive: the first organization keeps the existing databases and directory.
- **Upgrade every instance of a service together.** Some changes cannot be read by the previous version (for example a newly sealed key ring), and the upgrade notes say which changes cannot be rolled back past.
- **Turning on a feature is a step of its own.** Features that change keys, such as hardware-backed signing, are switched on only after every instance runs a release that knows about them, never in the same rolling upgrade.
- **Behavior changes are stated.** When a default changes (for example the session idle timeout, now on at 15 minutes), the upgrade notes say what users will see and how to keep the old behavior.
- **Stable deployment contract.** A test compares the shipped deployment files with recorded baselines, so a release cannot rename an image, port, setting or volume path unnoticed.

- **Moving people in.** People and groups can be loaded through SCIM 2.0. In the load test below, 60,000 test users were created through SCIM, one request each, in 2 minutes 15 seconds to 3 minutes 23 seconds, depending on what else the host was doing. The provisioning rate limit was raised for that load; at its default, one client creating that many accounts is slowed down.

EDCS has no numbered release yet; each delivery is identified by its build.

---

7. Monitoring and health checks

- **Liveness and readiness.** Every service answers a liveness check and a readiness check. The built-in probe runs inside the container without a shell. An HSM outage makes the sign-on service not ready while it stays live, so an orchestrator stops sending it signing work without restarting every instance.
- **Degraded, not down.** A stuck outbound provisioning message marks the provisioning check degraded, so a downstream outage cannot restart EDCS.
- **Audit integrity.** Each service offers an administrator-only check of its audit chain; chain faults are logged as errors. Alert on them.
- **Logs.** One line per request with method, path, status and time, plus every authentication, authorization and key-management message. Verbose per-statement logging is off by default.
- **SIEM export** of audit records and checkpoints, off by default; see the *EDCS Security and Standards Guide*.

---

8. Measured performance and sizing guidance

How it was measured

On November 9, 2026, one instance of each service, with its own database and directory server, ran on **one shared workstation-class host** (a 16-core, 32-thread desktop processor, 62 GiB of memory, an NVMe solid-state disk) that was also running other workloads, with the load generator on the same machine. The sign-on service was capped at 4 processors and 1 GiB of memory. The directory held 60,000 test users. The traffic was a **model** of 60,000 users, not a recording of a real site: everyone signs in once a day, 30% of sign-ins fall in the peak hour, 20,000 sessions refresh their tokens, applications ask for access decisions and read configuration, and the personnel feed changes accounts. A “sign-in” is the full browser flow: the sign-in page, the credentials and the code exchange. Sign-in limits, lockout, token lifetimes and audit were at their defaults.

Two settings differed from a default installation: the database connection limit was raised from 151 to 400 (with the default limit the spike would have run out of connections, so set it above the sum of the connection pools), and the provisioning rate limit was raised for the 60,000-account bulk load. The burst ran on the build that contains both fixes described below (November 9, 2026); later builds have not been load-tested. The other profiles ran on earlier builds, before one or both fixes.

Results

Profile	What happened
Weekday load, 11 minutes	47,379 requests, no errors. 95% of sign-ins within 104 ms; 95% of token refreshes within 75 ms. The sign-on service used 0.4 of one processor on average.
Sudden tenfold rise for 2 minutes	50 sign-ins, 200 access decisions and 50 configuration reads a second: no errors, 95% of sign-ins within 271 ms, back to normal afterward. Database connections rose from 89 to 321 and stayed there, which is why the connection limit was raised.
60,000 different users signing in within 15 minutes (66.7 a second, on top of background traffic)	232,914 requests, no failed request; 95% of sign-ins within 275 ms. 57 planned sign-ins (0.09%) were not started during one 3-second pause, so the test runner scored the run as a missed target, and the sign-on service's memory was at 818 of 1,024 MiB and still rising when the burst ended.
Weekday load, soak	Ran 24 minutes 37 seconds of a planned 30 before the host's own safeguard stopped it: 106,912 requests, no errors, latency flat; 4 planned requests were not started as the run was stopped. Memory grew about 9 MiB a minute and had not leveled off.
Stepped sign-in stress	95% of sign-ins within 69 ms at 20 a second and 135 ms at 40 a second; 1,080 ms in a partial 34-second step at 60 a second while other jobs loaded the host. 0 errors in 7,134 requests. The build before both fixes stalled at 80 a second (95th percentile 27,428 ms).

Two defects found during these runs were fixed and re-measured: a stall when every worker was waiting for a directory connection, and excessive per-request logging.

What these figures do not show

- **They are not a capacity guarantee.** One machine, one instance of each service, short runs.
- **Where it breaks is not known.** After the logging fix, the shared host stopped every stress attempt before the limit was reached, and no stress run was made on the build with both fixes.
- **Open items found in the data:** the sign-on service's memory growth (the largest open risk); a database lock that every token issuance for one organization passes through (part of the tamper-evident audit design); database cost per sign-in that rose during the burst; connection pools larger than the database's default connection limit.
- **Not tested under load:** several instances, smart-card, SAML or federated sign-in, second factors, TLS and a reverse proxy in the path, a government cloud, a run as long as 30 minutes, and the administration console.
- **Users behind shared addresses.** By default each client address may make 10 credential requests and 5 password attempts a minute. A site whose users reach EDCS through a few shared outbound addresses would meet those limits long before any capacity limit; plan the proxy and limits for it.

Sizing a deployment

We do not publish a sizing table: a table of processors and memory per user count would be a calculation from one machine, not a tested result. We size each deployment with you from these measurements, allow the sign-on service at least 2 GiB of memory until the open memory question is answered, and recommend running the same load suite on your own hardware before going live. We can help run it.

What is not included

- **No government authorization.** Neither EDCS nor the government-cloud template carries a Federal Risk and Authorization Management Program (FedRAMP) authorization, a Department of Defense (DoD) Impact Level authorization at any level, or an Authority to Operate (ATO). A government cloud region does not provide one by itself.
- **The government-cloud template has not been applied anywhere.**
- **No cross-site replication or failover** between data centers.
- **No validated cryptographic module of its own.** In Federal Information Processing Standard (FIPS) mode, EDCS relies on the validated module of the operating system it runs on. The standard images do not include a FIPS provider; a FIPS deployment uses images built on a FIPS-capable base. The procedure that creates and unlocks the vault's key-encryption key has not been reviewed for FIPS mode.
- **No published capacity guarantee.** The performance figures come from one workstation-class host.
- **No DoD enterprise-service connectors yet.**
- **No hosted software-as-a-service offering yet.** A dedicated installation we operate is arranged per engagement, on commercial servers, not in a government cloud region.
- **Not yet running in our own production.** Running in our own production: OAuth 2.0 and OpenID Connect sign-on service; Two-factor sign-in; Magic-link sign-in; Short-lived tokens; Directory with an API; Envelope-encrypted secrets vault; Central application configuration and feature flags; Per-application service clients; Audit events; Admin console; Administrator sign-in as a user (Quick Login); Multiple organizations, kept apart. Built and tested, shown on a test installation only: Browser single sign-on with PKCE; CAC and PIV smart-card sign-in; SAML 2.0 federation; Security keys and passkeys; Assurance levels and step-up; Sign-in through partner organizations; Sessions, token services and sign-out; Sign-in notice and consent; SCIM 2.0 provisioning; Outbound provisioning; Attribute-based access rules; Service clients without shared secrets; Entitlement record with attestation and reattestation; Separation of duties; Access requests with approval; Access decisions for your applications; Tamper-evident audit trail and SIEM export; FIPS mode on the platform's module; Signing keys in a hardware security module; Signing-key rotation; Several instances within one site; Delegated administration.

Contact

Questions about this guide, a sizing conversation, or a walkthrough on a test installation: curtilage.zone/contact.

Curtilage · Acknowledged Development Inc.