AI Evidence Harness

Access & Deploy — Job Aid

A product-neutral evaluation & evidence engine: pinned golden datasets, immutable recorded runs, dual-verifier scoring, and signed evidence reports — exposed as a versioned Evidence API and a console. AssureINS consumes it as the **harness_challenge_set** evidence method.

Read progressively. Level 1 is a 60-second skim. Level 2 is how to actually do it. Level 3 is the deep detail and the first-provision gotchas that cost real time — read it before you stand up a new environment.

Audience	FiveM engineers + operators accessing or deploying the harness.
Scope	Access (console + Evidence API) and deploy (per-environment pipeline).
Environments	dev · test · uat · prod (uat/prod are reviewer-gated).
System of record	PostgreSQL + row-level security. Foundry is stateless inference.

LEVEL 1

At a glance

Access — three steps

- **Console (people):** open the environment URL and **Sign in with Microsoft**. Your role comes from Entra group membership; pick a client engagement to scope everything you see.
- **Evidence API (services):** acquire an app-only Entra token for `api://<evidence-api-app>/.default`, carrying the **Evidence.Read** app role. Present it as `Authorization: Bearer <jwt>`.
- **What you get back:** run metrics and evidence-report summaries — the JSON headline AssureINS embeds, plus the signed report artifact.

Deploy — three steps

- **Merge to `main`.** CI must be green first (lint, types, tests, the full image build incl. `next build`, DB migrations + RLS).
- **GitHub Actions** logs into Azure via OIDC (no secrets) and runs the per-environment deploy: Bicep infra + ACR image build + container-app revision + an e2e smoke.
- **Promotion order:** dev → test automatically; **uat / prod are gated** on a required reviewer. Run `/milestone-review` before any uat/prod promote.

The golden rule of this platform: **Postgres + RLS is the system of record**; the Evidence API is a thin, versioned contract over it; Foundry is stateless inference. Nothing you access or deploy should violate that.

LEVEL 2 · ACCESS

Reaching the harness

A · The console (interactive)

Browse to the environment host and sign in with your FiveM Microsoft account. Authentication is Entra SSO (Authorization Code + PKCE); there is no local password. Your capabilities are resolved from Entra **security-group membership** — PLATFORM_ADMIN, MANAGER, MEMBER, or VIEWER — not from a table you can edit. Internal staff are platform-wide; the engagement switcher chooses which client's data (the RLS boundary) you are looking at.

B · The Evidence API (service-to-service)

A consumer (for example, the AssureINS web app) authenticates as its **managed identity** and calls the read endpoints. Two things must both be true, or the call fails closed:

- the calling identity is **granted the Evidence.Read app role** on the API app registration; and

- that identity is **registered as a consumer** (an `app_user` row) bound to exactly one client — that binding, not anything in the token, is the tenant boundary.

Acquire a token and call an endpoint:

```
# managed identity / DefaultAzureCredential, scope = the API audience
TOKEN=$(az account get-access-token \
--resource api://<evidence-api-app-id> --query accessToken -o tsv)

curl -H "Authorization: Bearer $TOKEN" \
https://<harness-host>/api/v1/runs/<runId>/metrics
curl -H "Authorization: Bearer $TOKEN" \
https://<harness-host>/api/v1/evidence-reports/<reportId>/summary
```

The token is validated against the tenant JWKS (RS256 only; issuer, audience and expiry all pinned). A validly-signed token that is not provisioned to a client returns **403**; a missing/expired/wrong-audience token returns **401**. This is by design — a signature proves identity, provisioning grants tenant access.

Key endpoints

GET /api/v1/runs/{id}/metrics	Aggregate metrics + the dual-verifier gate summary.
GET /api/v1/evidence-reports/{id}/summary	JSON headline (pass/gate/reproducibility) for embedding.
GET /api/v1/evidence-reports/{id}/download	The signed, immutable report artifact bytes.
GET /api/v1/subjects · /datasets · /eval-definitions	The catalog behind a run.

LEVEL 2 · DEPLOY

Shipping the harness

Deployment is fully GitHub-Actions driven and passwordless. A push/merge to `main` triggers the deploy workflow, which runs one job per environment in order.

Each environment job does:

- **OIDC login** to Azure using a federated credential bound to that GitHub environment — no stored client secret.
- **Infra:** `az deployment sub create` over `infra/main.bicep` with the env's `*.bicepparam` (parity is enforced by sharing one module set; only parameter values differ).
- **Images:** `az acr build` of the web + scheduler images, then the container-app revision is updated to the new tag.
- **Verify:** a post-deploy Playwright smoke (`e2e-dev`) runs against the live environment.

dev and **test** deploy automatically on a green build; **uat** and **prod** require a named reviewer to approve the environment. Migrations are forward-only and applied per environment during deploy.

LEVEL 3 · DEEP DETAIL

First-provision reality & gotchas

Read this before standing up a *new* environment or wiring a new consumer. Each item below cost real deploy cycles the first time.

Per-environment bootstrap is required — and per-environment

A green **dev** deploy does **not** mean test/uat/prod can deploy. Each environment needs its own one-time bootstrap: an Entra deploy app + federated credentials, the deploy identity granted **Contributor + User Access Administrator** on the resource group (it creates role assignments), a Key Vault auth seed, and the interactive-login Entra app + RBAC groups. A missing `roleAssignments/write` grant is the classic test-env failure.

OIDC subjects are immutable and numeric

GitHub issues federated-credential subjects for new repos in the numeric form `repo:OWNER@<ownerId>/REPO@<repoId>:environment:<env>`. Name-based federated credentials will fail with AADSTS700213 — create the credentials with the numeric subject.

Real auth vs the dev shim

The Evidence API validates real Entra JWTs and resolves the RLS client from the caller's `oid` via `app_user`. A dev-only bearer shim exists solely for local/e2e and is honored **only** when `NODE_ENV≠production` and `EVIDENCE_DEV_AUTH=1`; the deployed image forces `NODE_ENV=production`, so the shim is off in every real environment (fails closed). Turn real validation on by setting `EVIDENCE_API_AUDIENCE` + `EVIDENCE_ENTRA_TENANT_ID`.

Register a new API consumer

- Grant the consumer service principal / managed identity the **Evidence.Read** app role on the API app.
- Insert an `app_user` row binding its Entra `oid` to the client it may read. Dev Postgres is Entra-only, so use the temp-admin + firewall path: `az postgres flexible-server microsoft-entra-admin create` … (note: the subcommand was renamed from `ad-admin`), open a firewall rule for your IP, connect with an access token as the password, insert, then remove the rule.

Build gotchas that only surface at deploy

- **Extensionless relative imports.** Under Next's `moduleResolution: Bundler`, a `./foo.js` import of a `.ts` file resolves in `tsc` but breaks `next build`. Drop the extension.
- **Workspace dists must exist before** `next build`**.** Build the shared packages in dependency order first — CI now runs the full image chain so this is a red PR check, not a failed deploy.
- `.dockerignore` **must exclude** `*.tsbuildinfo` — a stale build-info file makes `tsc -b` skip emitting `dist` in a clean container build.

Inference (if/when a verifier model is wired)

Model access is Azure AI Foundry (resource kind `AIServices`, not the deprecated `OpenAI` kind), keyless via managed identity, `DataZoneStandard` SKU (US data zone). The harness has no LLM verifier today, so those deployments are idle.

This job aid is a living document — when a step changes, update the source and regenerate. For authoritative, versioned detail, the repository CLAUDE.md and docs/decisions/ ADRs win.