— ACT III - INTEGRATION - TEST DATA

Deterministic data is the bedrock under every test, demo, and dev server.

Last chapter we wired the SSE event contract. Now we step back to something quieter but just as load-bearing. Deterministic means the same seeder run always produces the same resource graph - same IDs, same references, same values. That is what lets a test assert on a known ID, gives a demo a stable patient roster, and makes every developer's local server identical.

The four disciplines that protect it

- Idempotent
- Phased
- Atomic
- Tenant-aware

— DISCIPLINE 1 - IDEMPOTENT SEEDING

Run the seeder twice - get one identical state, never a duplicate.

Ava

Nothing. Same state, no duplicates - that is idempotency. The footgun is the naive loop where every run adds another Patient.

Two ways to converge

An idempotent run leaves exactly one record.

— DISCIPLINE 2 - PHASE ORCHESTRATION

Seed the referenced before the referencing - every reference resolves.

FHIR resources point at each other - an Encounter references a Patient, an Observation references both Patient and Encounter, a Patient references a managing Organization. Seed in dependency order so every reference target already has a known ID.

Five phases, one dependency rail

5 Observation
References Patient and Encounter - both already seeded.

- Alternative - one transaction Bundle with urn:uuid forward-refs lets HAPI resolve intra-Bundle order (Ch 05)
- But phase staging keeps each batch small and debuggable

— DISCIPLINE 3 - ATOMIC BATCHES

Wrap each phase in a transaction Bundle - all entries commit or all roll back.

HAPI commits every entry in a transaction Bundle, or rolls the whole batch back on any failure - the same atomicity we built in Chapter 05. A loop of single REST calls can fail halfway and leave you torn; the transaction Bundle is the seed-batch primitive you want.

— BUNDLE.TYPE = TRANSACTION - IDEMPOTENT ENTRY

{ "fullUrl": "urn:uuid:pat-1", "request": { "method": "POST", "url": "Patient", "ifNoneExist": "identifier=mrn|001" } }

urn:uuid fullUrls carry the forward-references; If-None-Exist on each request makes a matching re-run a clean no-op - atomicity and idempotency combined.

— WHY NOT A LOOP OF CALLS

A loop half-applies on error -> torn state

Individual REST calls have no shared rollback. One failure mid-loop leaves some resources created and others missing - exactly the partial dup a transaction Bundle prevents.

— DISCIPLINE 4 - TENANT-AWARE SEEDING

Loop the clinics - one partition per batch, fixed IDs coexist.

Marcus - production scar

For each clinic slug, set X-Clinic-ID, then run the whole phase pipeline. The partition interceptor from Chapter 11 resolves the header to a partition ID.

Clinic loop -> partition isolation

The same seed-pat-001 exists once per tenant.

— RESET - A TRUE BLANK SLATE

A soft delete keeps history. A hard wipe starts genuinely empty.

Tests need a clean slate between runs - but a normal FHIR delete is a soft delete, and re-seeding the same fixed IDs then collides with retained version state.

— THE TRAP - SOFT DELETE

Tombstone, history retained

A normal DELETE marks a tombstone and keeps the history rows. Re-seed the same fixed IDs and you collide with retained version state.

● Re-seed collision

— THE CLEAN PATH - HARD WIPE

Expunge resources and their history

The delete-expunge operation hard-removes resources and their history rows by a search match, so re-seeding starts truly empty. Scope it to one partition - run under that clinic's X-Clinic-ID - to reset one clinic without touching siblings.

● Destructive - gated - dev/test only

It is destructive and gated behind an expunge config. The exact parameters and the enable key you confirm against your HAPI version.

— TWO SOURCES - ONE PIPELINE

Hand-curated fixtures for assertions; generated volume for everything else.

A typical pipeline uses both - a small fixture core the tests assert on, plus an optional generated bulk layer the tests merely tolerate.

— FIXTURES - HAND-AUTHORED

Fixed IDs, known clinical content

Version-controlled resource files with exact, recognizable values. Reach for these when a test asserts on a known ID, or a demo needs a stable roster.

● Assertion-grade

— GENERATED - SYNTHESIZED AT VOLUME

Synthea-style or a parameterized factory

Programmatic volume for load tests, search-result variety, and populated-UI screenshots. Determinism still applies - seed the random generator from a fixed value so volume is reproducible too.

● Fixed seed -> reproducible

— SYNTHESIS - THE CONCEPTUAL SEEDER SHAPE

All four disciplines on one rail - the pattern, not a literal tool.

Think of it as the Dudoxx seeder shape - conceptual, the pattern a seeder like ddx-fhir-seeder would follow. Outermost is the clinic loop; teardown lives beside it.

Clinic loop -> phase pipeline -> reset path

- Phased
- Atomic
- Idempotent
- Tenant-scoped

— TRY IT - RECAP - NEXT

Write a two-phase idempotent seeder - then run it twice.

— TRY IT YOURSELF

PUT /Organization/seed-org-001 -> then a transaction Bundle creating a Patient with If-None-Exist on the MRN

Run it twice and confirm the second run changes nothing - same IDs, no duplicates. Then wrap it in a clinic loop that sets X-Clinic-ID per tenant.

What you can now do

- Deterministic
- Idempotent
- Phased
- Atomic
- Tenant
- Reset
- Next - Chapter 17 - SMART on FHIR + RBAC