— CHAPTER 08 · PROFILING & VALIDATION · BEAT 1 — THE CONTRACT

The server is standing. Now — how do we say what counts as valid?

Resources flow in and out, but nothing yet enforces that in OUR system a Patient must carry an identifier, or an Observation must use a LOINC code. That is profiling. A profile constrains a base resource into a contract — and the validator is where the contract bites.

Three goals for this chapter

- Constrain — a base resource with a StructureDefinition
- Slice + must-support — split repeats, declare capability
- Validate — a real instance against a profile in HAPI

— CONCEPT · A PROFILE IS A CONSTRAINED BASE RESOURCE

A StructureDefinition with derivation constraint narrows a base.

Take Patient and tighten it. Three legal moves on one element — narrow cardinality within the base limits, fix a value, and re-bind terminology. Remember the four binding strengths from chapter four — you can only tighten, never weaken. A base required binding stays required.

— ELEMENTDEFINITION · THREE MOVES

Patient.identifier

"min": 1, "max": "1" — was 0..1, now required
"fixed[x]" / "pattern[x]" — element must equal exactly this
"binding": { "strength": "required" } — re-bound, tighter only

— THE HARD LIMIT

Constrain — never extend

A profile can narrow what already exists. It cannot rename a base element or add a brand-new one.
New elements need extensions — a different mechanism we met earlier. Profiling is subtraction, not addition.

- CAN narrow cardinality within base limits
- CAN fix or re-bind — strength only stronger
- CANNOT rename or add new elements

— DIAGRAM · SLICING A REPEATING ELEMENT

Slicing splits a repeat into named, mutually-exclusive lists.

Observation.component — sliced

discriminator: type value · path code

D slice: diastolic
component.code = 8462-4

- open — extras allowed
- closed — named only
- openAtEnd — extras after

Marcus & Ava

discriminator

How does the validator tell them apart?

— CONCEPT · THE DISTINCTION THAT TRIPS EVERYONE

Must-support is capability. Cardinality is presence.

The single biggest misconception in profiling — mustSupport true does not mean must-populate. They are two different mechanisms. Confuse them and you build the wrong validation, or fail a certification for the wrong reason.

— MUSTSUPPORT — CAPABLE OF HANDLING

Must-SUPPORT

The implementing system must be CAPABLE of handling the element — storing it, displaying it, processing it.
It says nothing about whether any given instance contains it. And a profile declaring must-support must also state what kind of support it expects.

— CARDINALITY — MUST BE PRESENT

Must-POPULATE

Cardinality forces presence. 1..1 means the element must actually appear in the instance. This is the trap — people read mustSupport and assume required. Presence is governed by min, not by support.

- mustSupport — implementer capability
- cardinality 1..1 — instance must contain it

— WALKTHROUGH · PROFILES YOU CAN ADOPT — US CORE

US Core profiles the core US clinical data set on FHIR R4.

In the US you rarely write from scratch — you adopt US Core. Three foundational profiles to know, each declaring the must-support elements a conformant system has to handle.

DEMOGRAPHICS

US Core Patient

Demographics and identifiers, with must-support elements a conformant system must handle.

must-support

A FAMILY OF PROFILES

US Core Observation

Not one profile — a family: vital signs, lab results, and screening and assessment.

must-support

SPLIT IN TWO

US Core Condition

condition-encounter-diagnosis and condition-problems-health-concerns.

must-support

Current — US Core 8.0.1 · STU 8 · 2026 (curriculum index written at v7; numbering stable, content current)

— CONCEPT · CROSSING BORDERS — THE INTERNATIONAL PATIENT SUMMARY

IPS is a portable summary built from a Composition.

US Core is American. When care moves between countries, the International Patient Summary — IPS — is the cross-border answer. Think of it as a portable document pulling together the themes that matter when a clinician has never seen you before.

— THE RECURRING SECTION THEMES

Allergies · Medications · Problems

A Composition-driven document that gathers a patient's key clinical context into a single, shareable summary.
We keep IPS at the concept level — know that it exists, and why it matters when care crosses a border.

— WHY THIS MATTERS IN THE EU

The cross-border payload format

IPS is the payload format behind Europe's cross-border health-data exchange — and the regulatory clock is real, with a phased deadline through the end of this decade.

- Portable patient summary
- Composition-driven document
- Cross-border / EU motivation

— WALKTHROUGH · HOW PROFILES REACH THE SERVER

A profile is just data until you load the package that carries it.

Profiles like US Core ship as an Implementation Guide package — NPM-style. Load it into the server we stood up, and the validator can finally reach the profiles.

— IG PACKAGE ANATOMY — PACKAGE.TGZ

package.json + conformance resources

StructureDefinitions · ValueSets · CodeSystems · examples
One tarball bundles everything the validator needs to enforce the guide.

— LOAD IT INTO HAPI

hapi-fhir-cli upload-definitions

./hapi-fhir-cli upload-definitions -t http://localhost:8080/fhir -d ./us.core.tgz
Or pre-load the package at startup. Either way, the profiles now live in HAPI's validation and terminology store.

- package.tgz — conformance bundle
- CLI upload or startup pre-load
- Profiles now reachable by the validator

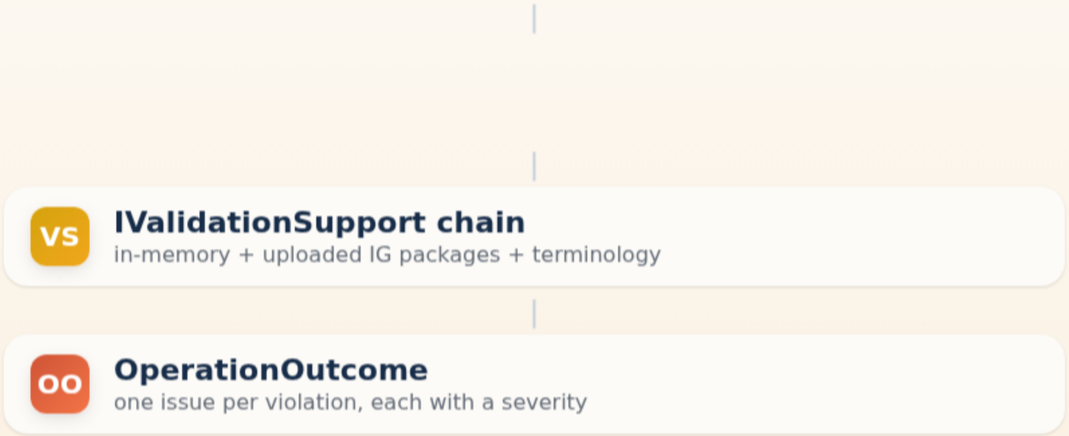
— FRAMEWORK ANCHOR · HOW HAPI CHECKS A RESOURCE

FhirInstanceValidator walks an instance against its profile.

You build a FhirValidator, give it the FhirInstanceValidator module, and back it with an IValidationSupport chain. Input — an instance plus the profile from meta.profile or an explicit URL. Output — an OperationOutcome, one issue per violation.

The validation chain

instance + profile to OperationOutcome



— WHAT EACH ISSUE CAN FLAG

The contract, enforced in code

Cardinality too low. A code outside a required binding. A slice that does not match. A missing must-support capability.
Each becomes an issue with a severity. That OperationOutcome IS your profile being enforced — not documentation, executable contract.

- cardinality
- binding
- slicing
- must-support

— CONCEPT · A CUSTOM ORG PROFILE — DERM4HAIR

Where the contract bites — at write time.

Derm4Hair Patient — three constraints

derivation: constraint · base: Patient

● Enforced via FhirInstanceValidator in the create path

Ava & Marcus

● write time

So where does this profile actually bite?

— CODE · SEE THE ACTUAL ARTIFACT

The differential is just your changes against the base.

Here is the StructureDefinition itself — compact, reviewable, and exactly what you ship inside the IG package.

— STRUCTUREDEFINITION · DERM4HAIR PATIENT

"derivation": "constraint" — a profile, not a new resource

"baseDefinition": "http://hl7.org/fhir/StructureDefinition/Patient"
"differential": { "element": [
{ "path": "Patient.identifier", "min": 1 } — now required
{ "path": "Patient.identifier.system", "fixedUri": "https://derm4hair.example/mrn" } — fixed system
{ "path": "Patient.communication.language", "mustSupport": true } — capability declared
] }

- derivation: constraint
- baseDefinition is Patient
- min narrowed to 1
- fixed[x] system
- mustSupport true

— CODE · TRY IT YOURSELF · 60 SECONDS

Validate a Patient against US Core — one call.

Post a Patient to the dollar-validate operation with a profile parameter pointing at US Core Patient, then read the issues. Severity and diagnostics tell you exactly what failed.

— RUN THIS AGAINST THE PUBLIC HAPI SANDBOX

```
curl -s 'https://hapi.fhir.org/baseR4/Patient/$validate?profile=http://hl7.org/fhir/us/core/StructureDefinition/us-core-patient' -H 'Content-Type: application/fhir+json' -d @patient.json | jq '.issue[] | {severity, diagnostics}'
```

One call, and you have validated against a real published profile — no SDK, no setup.

Recap

- Constrain a base resource
- Slice a repeating element
- Declare must-support
- Package profiles into an IG
- Enforce with the validator