— CHAPTER 12 · SUBSCRIPTIONS, BULK DATA, MDM · BEAT 1 — THREE FEATURES, ONE CHAPTER

So far the server answered when we asked. Now it acts on its own — at population scale.

Three features sit together this chapter. Subscriptions, so the server pushes you a notification when something changes. Bulk Data export, so you pull a whole population out asynchronously. And MDM — master data management — the golden record that decides two patient files are really one person.

What you will be able to do

- Subscriptions — criteria push + rest-hook vs message channel
- Bulk $export — async NDJSON at system, group, patient scope
- MDM — golden record, EID anchor, match scoring

— CONCEPT · THE R4 SUBSCRIPTION — TWO ESSENTIAL PARTS

A criteria search string, and a channel. That is the whole resource.

The criteria is just a search — the exact same search we built two chapters back. The channel is a type plus an endpoint. Match on create or update, and the server fires.

— SUBSCRIPTION — R4

{ "status": "requested", "criteria": "Observation?code=http://loinc.org|2823-3", "channel": { "type": "rest-hook", "endpoint": "https://my-app.example/hooks/labs", "payload": "application/fhir+json" } }

If you can write the search, you can write the subscription. That is the elegant part.

Three things to point at

- criteria — a search string · the same engine as Chapter 09
- channel.type — rest-hook here · message is the other option
- fires on create or update of a matching resource

— CONCEPT · THE R4 PAIN — POWERFUL PER-SUBSCRIBER, BUT NO SHARED TOPICS

Can I subscribe to a shared topic everyone reuses? In R4 — no.

Ava

Here is the scar — no. In R4 there is no SubscriptionTopic. No shared, curated topic abstraction at all.

Marcus · the three R4 limits

Every one. And criteria is single-resource-type — you cannot say Observation or DiagnosticReport in one. And it fires on create or update only — delete a resource and no notification goes out.

— COMPARISON · R5 FIXED IT — AND HAPI BACKPORTS THE FIX ONTO R4

R5 introduced SubscriptionTopic. HAPI brings it back to the R4 wire.

A SubscriptionTopic is a canonical, curated topic that defines the filters and the notification shape once, centrally. Subscribers just reference the topic — and you do not have to leave R4 to get it.

— R4 NATIVE · THE LIMIT

Each subscriber writes its own criteria string

No shared topic. Three teams watching the same event maintain three fragile strings — and they drift. Single-resource-type, no curation.

— R5-STYLE BACKPORT · THE FIX

Subscriptions reference one canonical SubscriptionTopic

Filters and notification shape defined centrally, via HAPI's Backport Subscription implementation guide. Same R4 messages on the wire — now with shared topics.

- **Version-sensitive — enabled in HAPI's topic-based subscription configuration**
- **The model is stable · check your HAPI version's docs for the exact setting**

— COMPARISON · CHANNEL.TYPE — THE TWO THAT MATTER

rest-hook for simple. message for serious.

The channel decides how the notification leaves the server. Two choices carry real weight; websocket and email also exist but are rarely what you want.

— REST-HOOK

Server POSTs to your endpoint

A synchronous HTTP POST to channel.endpoint, optionally with the full resource as payload. Simple, one consumer. Great for a webhook into a single service.

— MESSAGE

Delivered as a FHIR message to a broker

In HAPI this maps to a message-queue — a broker like Kafka. Durable, replayable, fans out to many consumers at once.

- **rest-hook — simple single-consumer webhook · down means the event is lost**
- **message — durable, multi-consumer, replayable fan-out**
- **websocket and email exist — rarely the right reach**

— DIAGRAM · BULK DATA V2 — THE $EXPORT OPERATION

That was events one at a time. Bulk moves a whole population.

Bulk Data version two is the dollar-export operation, at three scopes. Search answers a question; bulk moves an entire population to analytics, a payer, or a machine-learning job. Different operation class, different mechanics.

SCOPE · SYSTEM

/$export

Everything the server holds. The most expensive — bound it carefully.

SCOPE · GROUP

/Group/[id]/$export

A defined cohort — the usual choice for a population pipeline.

SCOPE · PATIENT

/Patient/$export

One patient compartment. The safest scope to experiment with.

Why bulk is not search — it moves whole populations, not query answers

Bound it with _type · _since · _outputFormat · _typeFilter

— WALKTHROUGH · BULK EXPORT IS ASYNC BY CONTRACT

A whole population cannot come back in one response. So it does not.

Four steps. There is no synchronous bulk GET — async or nothing.

STEP · 1 KICKOFF

Prefer: respond-async

You request the export with this header. You are asking the server to work in the background.

STEP · 2 ACCEPTED

202 + Content-Location

No data yet — a 202 Accepted and a polling URL in the Content-Location header.

STEP · 3 POLL

202 working · 200 done

Poll the URL. A 202 means still working — honor Retry-After. A 200 means done.

STEP · 4 DOWNLOAD

NDJSON manifest

The 200 body lists file links — one NDJSON file per resource type. Download them.

NDJSON — newline-delimited JSON · one resource per line, built for streaming

One file per resource type · no synchronous bulk GET exists

— CONCEPT · MDM — MASTER DATA MANAGEMENT

The same person, twice. MDM links them into one golden record.

Clinic A and clinic B each file the same human. MDM links those source records into one golden resource — the master. It is the one feature today that writes a new derived resource, not just reads or pushes.

SCORING

MATCH · POSSIBLE_MATCH · NO_MATCH

Incoming resources score into three buckets. A clear MATCH links automatically.

ANCHOR

EID

The Enterprise Identifier — a cross-system id saying this is the same person everywhere.

SURVIVORSHIP

Golden fields

Survivorship rules populate the golden resource's fields from the linked sources.

REVIEW

POSSIBLE_MATCH

An uncertain match surfaces for a human to confirm — never linked blindly.

- N source records link to ONE golden resource
- MDM writes a derived record — the only write-back feature here

— DIAGRAM · PUSH, PULL, AND IDENTITY — ONE COHERENT DATA FLOW

Picture one event. All three features wire together around it.

A new Observation is created. No new facts here — just the wiring that makes the chapter one story.

PUSH

Subscription fires

The matching Subscription fires through its channel — rest-hook or the broker. The event goes out the moment it happens.

PULL

$export cohort

A scheduled bulk dollar-export sweeps the same population into an NDJSON cohort. The whole set, on a schedule.

IDENTITY

MDM golden record

Threading through both — the golden record is the identity every consumer resolves to, so push and pull agree on who the patient is.

- Push — the instant notification
- Pull — the population sweep
- Identity — the deduplicated patient both rely on

— COMPARISON · TWO CAUTIONS THAT BITE IN PRODUCTION — THE SCARS

One silently drops events. The other quietly drains your budget.

These are the failures almost no tutorial names. Internalize both before you scale.

— CAUTION · SUBSCRIPTION DELIVERY RELIABILITY

The in-memory cache is not shared across instances

HAPI's subscription matching cache lives in memory, per node. Run more than one HAPI instance without a shared broker like Kafka, and notifications get silently dropped — no error, just missing events. If you scale out, you need the broker. This is the footgun we teased at the start.

— CAUTION · BULK EXPORT COST

A full-system $export is expensive

Full-system jobs run long, consume storage for the NDJSON files, and can time out your polling. Scope to a group or a patient, and bound it with _since and _type. Do not dollar-export the universe on a whim.

- **More than one node without a broker = dropped events (Ch 11 · Ch 21)**
- **Bound large exports — scope and _since are not optional at scale**

— TRY IT YOURSELF · THE ASYNC HANDSHAKE IN TWO COMMANDS

Kick off a $export, read the Content-Location, then poll it.

Start at the patient level. Read the polling URL from the response header, then GET it until a 200 returns the NDJSON manifest. If your sandbox does not have bulk enabled, run it against the local docker-compose HAPI from Chapter 07.

— RUN THIS IN YOUR TERMINAL

curl -i -H 'Prefer: respond-async' 'https://hapi.fhir.org/baseR4/Patient/$export' # then: curl -i '<Content-Location URL>' until it returns 200

The 202 hands you a polling URL; keep polling until 200 returns the list of NDJSON files. That is the whole Bulk Data contract, end to end.

Recap

- Subscriptions push
- Bulk $export pulls
- MDM reconciles identity
- 1 footgun — the non-shared subscription cache