RCM & Billing Decision Guide

Companion to the Billing docs.

Section 1: Use Case & Participants

1.1 What is your reimbursement model?

- Insurance / claims-based – you bill payers and reconcile their responses
- Patient self-pay / cash – patients pay you directly; no payer claim
- Value-based / capitation – per-member-per-month or risk-based
- Some combination

Why: Claims-based reads every lane; self-pay skips the Charge → Claim → Payer lane; value-based leans on Foundations and the Value-Based lane (3.11).

1.2 How much of the revenue cycle runs in Medplum vs. delegated?

- **Clearinghouse-direct** – Medplum runs the cycle; the clearinghouse is transport. You own scrubbing, denials, and remittance posting. The most complete documented path.
- **RCM-automation partner** – you send encounter and charge data and delegate the back end; scrubbing, payer routing, denials, resubmission, patient collections, and reporting run on the partner's platform. The integration surface is claim submission; the partner's billing tasks sync back into Medplum as `Task`s.
- **Documents only (no electronic claims)** – you don't run claims through a clearinghouse or RCM partner. Generate a CMS-1500 PDF for the

occasional paper/portal payer, or hand patients a superbill to self-submit for out-of-network reimbursement. Common for cash-pay and OON practices (see 1.1); also a per-payer fallback you can layer onto any engine above.

- **External Practice Management (PM) / billing system** – Medplum is the clinical source of truth; charges or claims hand off to an existing PM/billing system that owns the rest.

In one line: a clearinghouse moves your claims; an RCM platform runs your revenue cycle. Either way Medplum owns the front half (coverage, charge capture, claim assembly); the paths diverge on the back half.

Revenue-cycle stage	Clearinghouse	RCM partner	Documents only	External PM
Coverage & charge capture (3.1–3.5)	You build	You build	You build	You build
Eligibility (3.3)	You build	Partner platform	Manual	External
Claim assembly (3.6)	You build	You build	You build	External
Scrubbing / validation	You build	Partner owns	Light	External
Submission (3.6)	You build (837P)	Partner submit op	PDF / patient files	External
Responses & ERA posting (3.7)	You build (webhook + poller)	Partner owns; Tasks sync back	Manual	External
Denials & resubmission (3.8)	You build	Partner owns; Tasks sync back	Manual	External

Revenue-cycle stage	Clearinghouse	RCM partner	Documents only	External PM
AR & reporting (3.10)	You build	Partner owns	Manual	External

How to choose:

	Clearinghouse	RCM partner
You get	Full control; billing data stays as FHIR in Medplum	A payer-rules engine and in-house billers; far less to build
Trade-off	You build and maintain the entire back office	Recurring fees; less control; AR and reporting live on the partner's platform
Choose if	You have RCM expertise and want a fully owned cycle	You want to outsource the cycle

Paths can mix. The choice sets which Section 3 lanes you build (3.5–3.10) vs. read only for the boundary.

1.3 Who does the billing work, and is the patient part of it?

- Charge capture: clinicians at point of care, or back-office entry?
- Coding and claim work: dedicated coders/billers, or automated with exception review?
- Accounts receivable (AR) follow-up: who works denials, aging, and patient balances?
- Patient: do patients see estimates, statements, or pay online?

Why: drives charge-capture design (3.5), who works denials and AR (3.8, 3.10), and whether patient-facing billing is in scope (3.9).

1.4 Where does coding happen, and how automated is it?

- Auto-derived from clinical data (visit type, orders, procedures)?
- Coder-reviewed before the claim is built?
- Fully manual?

Why: drives the coding and review model in charge capture (3.5) and where scrubbing exceptions surface (3.6).

1.5 New build or replacing/extending, and what else must billing touch?

- If replacing: what billing/PM system, and what gaps drove the change?
- If new: is billing core to the product (you're building an RCM/billing product) or additive (a clinical app that also needs to bill)?
- What other systems must billing read from or write to (clinical/EHR source, PM, clearinghouse, accounting/GL, payment processor, payer portals)?

Why: surfaces integration boundaries and the source-of-truth split behind 1.2, 3.6, and 3.7. Core-to-product builds tend to own the cycle (a clearinghouse); additive builds tend to delegate (an RCM partner or external PM).

Section 2: Feature Scoping

Go through each row together. For each cell under Yes / No / Nice-to-have / Not sure, mark the customer's answer. The **§** *column points to the deep-dive subsection that covers each feature.*

#	Feature	§	Yes	No	Nice-to-have	Not sure
1	Coverage & payer modeling	3.1				
2	Patient financial account / running balance	3.2				
3	Real-time eligibility & benefits (270/271)	3.3				
4	Prior authorization (pre-service)	3.4				

#	Feature	§	Yes	No	Nice-to-have	Not sure
5	Patient cost / Good-Faith Estimates	3.4				
6	Charge capture (encounter-linked) & pricing / fee schedules	3.5				
7	Coding (CPT/ICD-10, modifiers) with review queue	3.5, 3.6				
8	Assemble, validate & submit claims	3.6				
9	CMS-1500 PDFs / superbills (manual / patient submission)	3.6				
10	Ingest payer responses & post remittance (277CA / 835)	3.7				
11	Denials, corrected claims & appeals	3.8				
12	Patient statements, invoices & online payment	3.9				
13	Claim / AR status, aging & reporting	3.10				
14	Value-based contracts (panel, quality measures, capitation)	3.11				

Section 3: Feature Deep Dives

Cover each feature flagged Yes or Nice-to-have in Section 2. The goal is to land on a clear recommended approach by the end of each section.

Section 3 is grouped into four lanes:

Lane	Subsections	When to read
Foundations	3.1 – 3.2	Always
Pre-service / Patient Access	3.3 – 3.4	When you verify coverage or authorize before service
Charge → Claim → Payer	3.5 – 3.8	When you bill insurance
Patient & Operations	3.9 – 3.10	Always
Value-Based	3.11	When you have capitated or risk-based contracts

Foundations

3.1 Coverage & Insurance Model

How you represent a patient's insurance (the coverage stack, the subscriber, payers, cost-sharing) is the foundation every claim and eligibility check builds on. See Representing Patient Insurance Coverage; coverage capture at intake is in the Intake guide (3.5).

Questions:

- Do patients commonly have more than one coverage, and must you bill them in order?
- Is the subscriber sometimes someone other than the patient?
- Is your payer list curated in Medplum, synced, or created on demand?
- What cost-sharing (copay, coinsurance, deductible) do you need for estimates and patient billing?

Situation	Approach
Subscriber is the patient	One `Coverage`; `subscriber` = `Patient`; `relationship` = `self`; `payor` → `Organization`.
Primary + secondary	One `Coverage` per plan; `Coverage.order` sets billing sequence. Secondary submits after the primary remittance posts (3.7).
Subscriber ≠ patient	`Coverage` + `RelatedPerson` for the subscriber. The relationship inverts: `Coverage.relationship` is the patient's relationship to the subscriber, the `RelatedPerson`'s is the reverse (coverage `child` → related person `parent`). Getting this backwards is a common billing bug.
Member ID & plan details	`Coverage.subscriberId` for member ID; `Coverage.class` for group, plan name, RxBIN/RxPCN.
Cost-sharing	`Coverage.costToBeneficiary`; feeds estimates (3.4) and patient responsibility (3.9).
Payer directory	Medplum ships a built-in **Payer Directory** of `Organizations`, or curate your own. The payer identifier system differs by backend; see 3.6.
Card images	Store as `DocumentReference`/`Binary` linked to the `Coverage`.

3.2 Account & Financial Ledger

Decide whether you track a patient financial account that rolls charges, claims, and payments into a balance (the backbone for AR in 3.10 and patient billing in 3.9), or let each charge stand alone.

Note: the rows below are one FHIR approach to a financial ledger; validate against your reporting and reconciliation requirements.

Questions:

- Do you need a single balance rolling up charges, payer payments, adjustments, and patient payments?
- Is the account scoped per patient, per visit, or per episode?
- Do you track payer and patient responsibility separately on the same account?
- Must the account reconcile against an external accounting/GL system?

Situation	Approach
Running balance	`Account` as the ledger anchor; `ChargeItem`, `Claim`, `PaymentReconciliation`, and `Invoice` reference it.
Account scope	`Account.subject` → `Patient`, or tie to `Encounter` for visit-level; apply one scope consistently.
Payer vs. patient	Payer side via claim/remittance (3.7), patient side via `Invoice` (3.9), both rolling to the same `Account`.
No ledger	Skip `Account` for pure pass-through submission where another system owns the balance, but you lose in-Medplum AR (3.10).

Pre-service / Patient Access

3.3 Eligibility & Benefits

Decide whether you verify coverage before service, what you ask the payer, and when the check runs. See Insurance Eligibility Checks.

Note: Eligibility is provided by the clearinghouse integration. An RCM partner may offer its own eligibility on its platform, outside that integration.

Questions:

- What must you confirm: active coverage, covered service, in-network status, or benefit/cost detail?
- When does the check run: scheduling, check-in, or batch pre-visit?
- Do you re-check on a cadence for ongoing patients?

- Who works failed or ambiguous checks?

Situation	Approach
Verify coverage / benefits	`CoverageEligibilityRequest` (`purpose: benefits`); invoke the clearinghouse eligibility operation to run the X12 270/271 and return a `CoverageEligibilityResponse`.
Scope limit	The current integration returns general benefits only, not arbitrary service-specific benefit queries.
In-network check	Read network participation from the response (in-network indicator Y/N/U/W); run at provider selection or as a pre-schedule gate.
Cost-sharing	Copay, coinsurance, remaining deductible, and out-of-pocket land in `CoverageEligibilityResponse.insurance.item`; the input to patient estimates (3.4).
When it runs	A `Bot` issues the request on the triggering event; persist the response for the visit and billing.
Exception	On failure or ambiguity, create a `Task` for registration rather than proceeding silently.

3.4 Prior Authorization & Estimates

Decide whether services need payer authorization before they happen, how you track that to a decision, and whether you produce patient cost estimates up front.

Note: the eligibility- and Claim-based rows below are one FHIR approach to prior auth; validate against your payer requirements. The standards-based CMS-0057-F path (CDS Hooks / CRD) is in alpha. Prior auth in a referral context is in the Referrals guide (3.7).

Questions:

- Do any services require prior authorization before scheduling or delivery?
- How do you check whether auth is required, and track a submitted auth to approval/denial?
- Must scheduling be gated until authorization clears?
- Do you owe patients a cost estimate before service (e.g. a Good-Faith Estimate)?

Situation	Approach
Is auth required?	`CoverageEligibilityRequest` with `purpose: auth-requirements`; the response indicates whether authorization is required.
Submit a prior auth	`Claim` with `use: preauthorization` (distinct from a claim for completed services); `ClaimResponse` carries the decision. A `Bot` converts FHIR ↔️ X12 for transmission.
Standards-based (CMS-0057-F)	Electronic prior auth via CDS Hooks / Coverage Requirements Discovery (CRD); alpha. The emerging payer-interop path as the rule phases in.
Gate scheduling	Hold in an auth-pending `Task.businessStatus` until the `ClaimResponse` approves, then release.

Good-Faith Estimates (No Surprises Act). One FHIR approach: build the estimate from pricing rules (`ChargeItemDefinition`/`$apply`, 3.5) and the patient's cost-sharing (`Coverage.costToBeneficiary`, 3.1), delivered as a `DocumentReference`; validate against your compliance requirements. Mainly for self-pay and out-of-network patients.

Charge → Claim → Payer

Insurance-claims-specific. Self-pay builds skip to Patient & Operations; pricing (3.5) still applies to self-pay rates and estimates.

3.5 Charge Capture, Coding & Pricing

Decide what gets billed, where the codes come from, and how each line is priced. Coding is the largest source of denials, so treat it as first-class, not a byproduct of charge entry.

Questions:

- What generates a charge: a completed encounter, an order/procedure, a fulfilled service?
- Where do codes come from (CPT/HCPCS, ICD-10, modifiers): auto-derived, coder-reviewed, or manual (1.4)?
- Is pricing a single fee schedule, or does it vary by payer contract?
- Who reviews coding before a claim is built, and how do exceptions surface?

Situation	Approach
What gets billed	One `ChargeItem` per billable line; `ChargeItem.code` carries CPT/HCPCS. Link `context` → `Encounter` and reference the `Account` (3.2) at creation, or reconciling into claims and AR later gets painful.
Pricing	`ChargeItemDefinition` holds the rules; `$apply` calculates the price.
Payer-specific rates	Multiple `ChargeItemDefinitions` keyed to payer/contract; apply the one matching the coverage.
Coding source	Auto-derive via a `Bot`, or set by a coder; route low-confidence items to a review `Task`.

Situation	Approach
Diagnoses	Capture ICD-10 diagnoses to attach to the claim (`Claim.diagnosis`, with item-level pointers) in 3.6.

3.6 Claim Assembly & Submission

Decide how you build a valid claim from charges and coverage, and how it leaves your system. The 1.2 engine decision plays out here.

⚠️ **One-way door:** Choosing a submission backend is hard to reverse. Each maps the `Claim` to a different format (837P, the partner's API, a CMS-1500 PDF) with a different response pipeline (3.7), and payer identifier systems differ. Keep one `Claim` as the source of truth so the backend stays swappable.

Questions:

- Are you billing as an organization or an individual provider?
- Which submission path(s), and does it vary by payer (electronic for most, paper/portal for some)?
- What validation must pass before submission, and where do failures surface?
- Do you submit secondary claims after the primary pays (coordination of benefits)?

Situation	Approach
Build the claim	`Claim` referencing the `Coverage` (`Claim.insurance`), charges (`Claim.item`), and ICD-10 diagnoses (`Claim.diagnosis`) with item-level pointers.
Billing provider	Set on `Claim.provider`, but backends read it differently (billing provider vs. rendering provider with the billing `Organization` via `PractitionerRole`); confirm per integration.

Situation	Approach
Electronic (you own the cycle)	The clearinghouse submit operation sends an 837P and writes a correlation id back to the `Claim`.
RCM partner	The partner's submit operation returns a `ClaimResponse`; the partner owns scrubbing, denials, resubmission, and remittance on its platform, and billing tasks sync back into Medplum as `Tasks` (3.8).
Manual / paper / portal	`Claim/$export` generates a CMS-1500 PDF (`Binary`; experimental); log the outbound document.
Patient-submitted (out-of-network)	Generate a superbill for the patient to file.
Validation	Validate before submit; surface a `Task` and don't transmit invalid claims. (In the current integration, a submission failure marks the `Claim` `status = error`.)
Multiple channels	One `Claim` is the source of truth; run separate send paths per payer without diverging content.

3.7 Payer Responses & Remittance Posting

Decide how you ingest what the payer sends back (the *277CA* acknowledgment, accepted or rejected pre-adjudication, and the *835 ERA* remittance, what was paid, adjusted, or denied line by line) and how payments post. These share one pipeline, so design them together.

Note: applies to the clearinghouse path, and the row specifics reflect the current integration. With an RCM partner, the partner owns response handling and posting; what flows back is its billing `Tasks`.

Questions:

- How do responses arrive: webhook push, polling, or partner API?
- How do you match a response to the originating claim?
- How do payer payments and adjustments post to the account?
- What share posts automatically vs. needs review?

Situation	Approach
Ingestion	A webhook bot (primary) plus an optional poller (catch-up). Make it idempotent on the payer's inbound transaction id (not the webhook delivery id); webhooks redeliver, and without dedup you double-post payments.
Acknowledgment (277CA)	Persist as a `ClaimResponse`; surface front-end rejections to denials (3.8).
Remittance (835 ERA)	Create a `PaymentReconciliation` and `ClaimResponse`; the ERA PDF is stored as a `DocumentReference`. Post payments/adjustments against the `Account` (3.2).
Matching	Match on the patient-control-number (PCN) assigned at submission from the `Claim` id (835), or the correlation id (277).
Claim status	Status lives on `ClaimResponse`; query `ClaimResponse?request=Claim/{id}` (3.10).
Review	Auto-post clean remittances; route mismatches to a `Task`.

3.8 Denials, Rejections & Resubmission

Decide how you work claims the payer rejects or denies: correcting and resubmitting, or appealing.

Note: The rows below are one FHIR approach to denial handling; validate against your workflow. On the clearinghouse path you build this in Medplum; with an RCM partner, denials are worked on the partner platform and surface back as synced `Tasks`.

Questions:

- Do you work front-end rejections (277CA) differently from adjudicated denials (835)?
- When you fix and resend, is it a corrected claim or a fresh submission?
- Do you track appeals (deadlines, supporting documentation, outcome)?
- How do staff find claims that need work?

Situation	Approach
Surface denials	Denial detail lives on the `ClaimResponse` (3.7); create a `Task` work queue for the biller.
Corrected claim	Resubmit a corrected `Claim` linked to the original (`Claim.related`), via the same backend (3.6).
Appeals	Track as a `Task` with deadlines; attach supporting docs as `DocumentReference`.
Find work	Search over open denial `Tasks` (3.10).
Partner-owned	The RCM partner works denials/resubmission on its platform; surfaced tasks sync into Medplum as `Tasks`.

Patient & Operations

3.9 Patient Responsibility & Self-Pay

Decide how you bill patients for their share (copays, coinsurance, deductibles, full self-pay) and how they pay.

Questions:

- What balances do you bill: post-adjudication responsibility, self-pay, or both?
- Do you issue statements, and on what cadence?
- Do you take online payment / card on file, through which processor?
- For out-of-network patients, do you provide a superbill instead of billing the payer?

Situation	Approach
Patient balance	`Invoice` for patient-owed amounts rolling to the `Account` (3.2); responsibility from remittance (3.7) or self-pay pricing (3.5).
Online payment	Integrate a payment processor; a sample integration shows keeping invoices and payments in sync.
Self-pay / cash	Price via `ChargeItemDefinition/$apply` (3.5); invoice the patient directly, no payer claim.
Out-of-network	Provide a superbill (3.6) for the patient to submit for reimbursement.

3.10 Accounts Receivable, Aging & Lifecycle Status

Decide how you track where every claim and balance sits, and how staff find work that's stuck or overdue.

Note: The rows below are one FHIR approach to AR tracking; validate against your reporting needs. Most relevant on the clearinghouse path; an RCM partner owns AR and reporting.

Questions:

- What lifecycle states matter (e.g. draft → submitted → accepted → paid → denied → closed)?
- How do you standardize states so queues, reports, and integrations agree?
- Do you track aging (days since submission, days pending) and AR buckets?
- How do staff find overdue or stuck claims and balances?

Situation	Approach
Lifecycle states	Track progress with `Task.businessStatus`; move `Task.status` on clear transitions. Keep the state set small and single-sourced.
External signals	`Bots`/subscriptions advance status when responses arrive (3.7): accepted, paid, denied.
Aging & AR	Search over `Claim.created`, `Task.lastModified`, and `Account` balances; bucket by age.
Overdue work	Scheduled `Bots` or subscriptions flag items past an SLA into work queues.

Value-Based

3.11 Value-Based & Capitation

Value-based care isn't a claims pipeline; it splits into measurement (quality, gaps in care) and money (capitation, risk, settlement). Read only for capitated or risk-based contracts.

Note: Quality measurement uses documented operations; the financial rows are one FHIR approach to validate against your contracts. Quality/HEDIS reporting overlaps clinical analytics; see Analytics.

Questions:

- Which contracts are capitated or risk-based, and which members are attributed to you?

- Which quality measures (eCQM, HEDIS, stars) are you accountable for?
- Do you still submit claims or encounter data when capitated (for risk adjustment and measurement)?
- How are capitation payments and any shared-savings settlement reconciled?

Situation	Approach
Attribution / panel	`Group` as the member roster; ingest the payer roster via a `Bot`; `CareTeam` for accountable teams.
Quality measurement & gaps in care	`Measure` + `MeasureReport` via `Measure/$evaluate-measure`; eCQM and HEDIS via Analytics.
Risk adjustment (HCC / RAF)	Capture `Condition` coding from encounters and submit as encounter data; ties to coding (3.5).
Encounter data	Even when capitated, reuse claim assembly (3.6) to send 837/encounter data for measurement and risk.
Payments (PMPM, settlement)	Model received capitation and shared-savings settlement as `PaymentReconciliation` against the `Account` (3.2).

Related Integrations

The billing and revenue-cycle partners Medplum integrates with first-party:

- **Stedi** — X12 eligibility checks (270/271) and professional claim submission (837P) with claim responses.
- **Candid Health** — revenue cycle automation and professional claim submission from FHIR `Claim` resources.