Intake & Registration Decision Guide

Companion to the Intake & Registration docs.

Section 1: Use Case & Participants

1.1 Who completes intake, and in what setting?

- Patient self-service (portal, mobile, kiosk)
- Staff-assisted (registration desk, rooming, phone, paper-to-digital)
- Mixed (patient starts online, staff verifies in person)

Why: determines the auth model and whether an intake response may exist before the patient record is created (3.1, 3.2).

1.2 When does intake happen relative to identity and scheduling?

- Before the patient record exists (greenfield / new patient signup)
- After scheduling, pre-visit (patient exists, no visit yet)
- At check-in or during the visit
- Episodic updates only (existing patient, not tied to a visit)

Why: drives whether intake creates or updates a patient record, when responses get tied to a patient, and whether intake tasks link to a visit (3.1, 3.8).

1.3 How are returning patients and duplicates handled?

- Always create a new patient record on intake (rare; only if you have a separate dedup pipeline downstream)
- Look up by identifier (e.g. MRN, email, phone) and update if found

- Run a demographic candidate search and confirm with staff before merge
- Mixed – different rules for patient self-service vs staff-assisted

Why: drives 3.1 and must align with your org-wide dedup policy.

1.4 What consents need to be captured at intake, and are any state- or population-specific?

Why: drives 3.6; sensitive categories may constrain what post-intake automation can auto-share or auto-route (3.7).

1.5 What other systems must intake feed or read from?

- Eligibility / insurance verification (e.g. Stedi, clearinghouse 270/271)
- Practice management or legacy EHR (write-back of demographics or coverage)
- Prefill sources (Patient Access API, Payer-to-Payer API, partner EHR, HIE/TEFCA) — see 3.3
- Referrals (intake initiated by an inbound referral – see the Referrals guide)

Why: surfaces integration boundaries that affect prefill (3.3), post-intake automation (3.7), and which actions live inside one intake workflow vs separate workflows.

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 **Deep dive** *column points to the Section 3 subsection that covers each feature.*

#	Feature	Deep dive	Yes	No	Nice-to-have	Not sure
1	Multiple intake forms by visit type, program, or language (vs one universal form)	3.2				
2	Separate vs shared questionnaires for patient self-service and staff-assisted intake	3.2				
3	Non-form intake modality (conversational chatbot, voice agent) feeding the same data shape as the form	3.2				
4	Patient identity handling at intake — identifier lookup, demographic match search, and pre-auth start linked to a patient record later	3.1				
5	Prefill intake from prior data, payer/HIE/EHR sources, or insurance card OCR; reconciliation when patient input disagrees with the source	3.3				
6	Insurance capture with card images, multiple plans, and non-patient subscriber (e.g. parent's plan covering a child)	3.5				

#	Feature	Deep dive	Yes	No	Nice-to-have	Not sure
7	Capture clinical history (allergies, medications, conditions, SDOH, immunizations, family history) and persist as distinct structured records	3.2, 3.4				
8	Multiple consent types captured at intake, with renewal triggers	3.6				
9	Post-intake automation: notify staff, run eligibility, create exception tasks, route to a care team	3.7				
10	Multi-step or multi-author intake, with steps that appear in the provider's visit chart	3.8				
11	Treat "intake complete" as a tracked operational status that gates downstream work (scheduling, clinical)	3.9				

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 three lanes:

Lane	Subsections	When to read
Capture	3.1 – 3.4	Always
Coverage & Consent	3.5 – 3.6	When coverage or consent is in scope
After Submission	3.7 – 3.9	Always

Capture

3.1 Patient Identity at Intake

Decide whether intake creates a new patient record, updates an existing one, or runs unlinked until a human confirms a match – and how the intake response gets tied to the right patient. For matching rules, master records, and merge mechanics, defer to Patient Deduplication Architectures and [Patient `$match`](/docs/api/fhir/operations/patient-match).

Questions:

- Does the patient record exist before intake is filled out, after, or sometimes either?
- Do you run an identifier lookup, a demographic match search, both, or neither before creating a new patient record?
- When a match is ambiguous, does the intake response stay unlinked until a human confirms, or does the system pick the best candidate automatically?
- For self-service flows, can a patient submit intake before authenticating, and how is that response later associated with an account?

Situation	Approach
Patient exists before intake (scheduled visit, returning patient)	Set `QuestionnaireResponse.subject` at create; intake Bot updates the existing Patient.

Situation	Approach
New patient, no candidate match	Intake Bot creates Patient, then back-fills `QuestionnaireResponse.subject`.
New patient, demographic search returns candidates	Use Patient `$match` for built-in scoring; require human confirmation in a Task before linking.
Anonymous / pre-auth start	Allow QuestionnaireResponse without `subject`; link at the next authenticated step or staff review. Track unlinked responses for cleanup.

3.2 Form Library & Capture Flow

Decide whether intake uses one universal form or several, whether the same form serves both patient self-service and staff-assisted flows, and how non-form modalities (chat, voice) feed into the same data.

Questions:

- One intake form for everyone, or different forms by visit type, program, or language?
- Same form for patient self-service and staff-assisted, or separate forms tuned to each audience?
- If you offer conversational or voice intake, does the transcript land in the same data shape as the form, or in a separate pipeline?

Situation	Approach
One form for everyone	Single `Questionnaire`; one Bot maps to one consistent set of FHIR resources.
Forms by visit type, program, or language	Multiple `Questionnaires` with shared `linkId` conventions; Subscription criteria can target specific `questionnaire` URLs (see 3.7).

Situation	Approach
Patient vs staff variants	Either reuse one Questionnaire with a "completed-by" gate, or maintain parallel Questionnaires that map to the same target resources.
Conversational or voice intake	Keep a `Questionnaire` as the authoritative schema for what's collected; the chatbot or voice agent maps transcript fields to `linkIds` before submitting a `QuestionnaireResponse`. Downstream extraction (3.4) stays the same.

3.3 Prefill & Reconciliation

Decide whether intake starts blank or arrives pre-populated, where prefill data comes from, and how patient corrections are reconciled with the upstream source. Prefill is a superset of HIE — HIE is one source among many.

⚠️ **One-way door:** Once intake silently overwrites existing records from a prefill source, distinguishing "patient said X" from "source said X" after the fact requires per-field origin tracking. Set the origin-tracking pattern before launch; retrofitting historical responses is painful.

Questions:

- Does the form start blank, prefilled from data you already hold (returning patient, prior visit), or prefilled from an external source?
- Which external sources are in scope (insurance card OCR, Patient Access API, Payer-to-Payer API, partner EHR, TEFCA/QHIN, state or regional HIE)?
- When prefilled data and patient input disagree, who wins — the source, the patient, or staff?
- Does the patient see prefilled values and explicitly confirm, or is prefill silent?

Situation	Approach
Source selection	Choose any combination of: internal (returning patient / prior intake), insurance card OCR, patient-mediated FHIR pull (Patient Access API / SMART-on-FHIR / partner EHR), Payer-to-Payer (CMS-0057, Jan 2027), HIE / TEFCA / QHIN. Each lands as draft resources for reconciliation.
Conflict resolution policy	Pick one per field domain: source-wins (administrative), patient-wins (clinical history), last-write-wins, or prompt staff. Document it; don't leave it per-field.
Provenance	Create a `Provenance` resource per prefilled field (or batched per source pull) so downstream consumers know origin and pull date.
Patient visibility and refresh cadence	Default to "show and confirm" for clinical and demographic fields; silent prefill is acceptable for low-risk administrative fields when audit is in place. One-shot pull at form open is the safest default; re-pull on submit only for high-volatility sources (e.g. active coverage status).

3.4 Extraction Model – SDC vs Bot

Decide where extraction logic lives: on the form itself (declarative SDC rules) or in custom code – and how you handle resubmissions without creating duplicate records.

⚠️ **One-way door:** Once a form is in production and its field identifiers are referenced by extraction code or templates, renaming them requires migrating historical responses or maintaining a translation layer. Decide naming conventions before launch.

Questions:

- Does extraction need conditional logic (only create a subscriber/guardian record when the subscriber is not the patient, only create a consent record when the patient agreed)?
- Does extraction need to call external APIs (eligibility, address validation, geocoding) during processing?
- When intake is resubmitted (annual update, correction), what should happen to existing allergies, medications, conditions, and coverage records?
- Will non-engineers update the form, and how often?

Situation	Approach
Simple field-to-resource mapping, no conditionals, no external calls	SDC `$extract` with template resources in `Questionnaire.contained`; rules ship with the form.
Conditional resource creation, external API calls, custom validation	Bot triggered by Subscription on `QuestionnaireResponse`; use `getQuestionnaireAnswers` and `getGroupRepeatedAnswers` from `@medplum/core`.
Mixed	SDC for the deterministic mapping, Bot for the conditional / external pieces.

Situation	Approach
Resubmission without duplicates	Upsert clinical history by stable natural keys: `patient` + `code` for AllergyIntolerance; `subject` + `code` for MedicationRequest / Condition; `beneficiary` + `payor` for Coverage. See Intake Data Model → Resource Role Reference.

Coverage & Consent

3.5 Coverage & Subscriber Relationships

Decide how insurance is modeled, especially when the subscriber is not the patient, and how payers are sourced (curated directory or created on demand).

⚠️ **One-way door:** On the coverage record, the relationship field describes the **patient's** relationship to the subscriber; on the linked subscriber/guardian record, the relationship field describes that person's relationship to the patient – they invert (if coverage says "child", the subscriber is "parent"). Encoding this incorrectly is one of the most common intake bugs and is hard to retroactively fix on historical data.

Questions:

- Are multiple coverages (primary / secondary) captured at intake, or only primary?
- Is the subscriber sometimes someone other than the patient (e.g. a parent's plan covering a child)?
- Do you capture insurance card images, and where do they live?
- Are payers pre-loaded as a curated directory, or created on demand?

Situation	Approach
Subscriber is the patient	Single `Coverage` with `subscriber` = `Patient`; `relationship` = `self`.

Situation	Approach
Subscriber is not the patient	`Coverage` + `RelatedPerson`; remember the **inversion**: if `Coverage.relationship` is `child`, `RelatedPerson.relationship` is parent. See Intake Data Model → RelatedPerson and Insurance Coverage.
Multiple coverages	One `Coverage` per plan; use `order` to indicate primary vs secondary.
Card image capture	Store the image as a `DocumentReference` and link it to the Coverage via `DocumentReference.context.related`; for full modeling see Patient Insurance.
Payer directory	Curated `Organization` resources searched by the form; fall back to "create if missing" only when scope is too broad to curate.

3.6 Consents

Decide which consent types are captured, how they're modeled, and what triggers re-consent.

Questions:

- Which consent types are required at first intake vs deferred (HIPAA, treatment, financial, telehealth, research)?
- Are some consents conditional on patient attributes (minor, state of residence, payer)?
- What triggers re-consent (annual update, policy change, new clinic site)?
- Does intake need to capture electronic signature artifacts (image, attestation timestamp)?

Situation	Approach
One consent per type	One `Consent` resource per type captured (HIPAA, treatment, financial, etc.); don't bundle multiple agreements into one Consent.
Conditional consents	Use `enableWhen` on the Questionnaire; SDC or Bot creates Consent only when the patient agreed.
Re-consent	Existing Consent moves to `status: inactive`; new Consent created with current effective period.
Signature artifact	Store as `Binary` referenced from `Consent.sourceAttachment`.

After Submission

3.7 Post-Intake Automation – Triggers & Actions

Decide what happens after intake is submitted: which automations fire, when eligibility runs, and which downstream actions are part of intake vs separate workflows.

Note: This section is decision-focused. For the full pattern catalog, see Post Intake Automation.

Questions:

- Does one workflow do everything (extract + notify + verify eligibility + create exception tasks), or do you split concerns into multiple workflows wired to different triggers?
- Should intake processing re-run when the response is updated, or only on first submit?
- When does eligibility verification run – during form completion, on submit, or async after intake is committed?
- Which post-intake actions are required to mark intake "done" vs nice-to-have side effects?

- When eligibility, address validation, or another external call fails, does that block intake or surface an exception task?

Situation	Approach
Monolithic intake Bot	One Bot on `QuestionnaireResponse` create handles extraction and all downstream actions. Simplest to operate; trade-off is larger blast radius on failure.
Split by concern	Separate Subscriptions: one on `QuestionnaireResponse` for extraction, one on `Coverage` for eligibility, one on `Patient` for welcome notifications. Each Bot has a single responsibility and audit trail.
Prevent re-run on updates	Use the create-only Subscription extension so Bot updates to the QuestionnaireResponse (e.g. setting `subject`) don't re-trigger extraction. See Avoid Infinite Loops.
Exception handling	When required data is missing or an external call fails, create a `Task` for staff (`status: requested`, `owner` = registration pool); do not silently drop the response.
Eligibility — sync gate (during form or on submit)	`CoverageEligibilityRequest` results gate "intake complete" (3.9). Highest UX cost, lowest downstream rework. Use when day-one billing readiness matters.

Situation	Approach
Eligibility — async post-intake	Same Subscription pattern, but failures surface as exception Tasks rather than blocking intake. Use when intake throughput trumps day-one billing readiness.
Care-team / program routing	Bot creates Tasks or CareTeam memberships based on intake answers (e.g. screening result triggers a behavioral health referral).

3.8 PlanDefinition Orchestration & Encounter Linkage

Decide whether intake is a single form or a multi-step workflow with several tasks owned by different roles – and whether those tasks must show up in the provider's visit chart.

Note: The template-driven orchestration pattern is documented in detail under Post Intake Automation → PlanDefinition Orchestration. This section stays at the level needed to decide whether to adopt that pattern.

Questions:

- Is intake completed by one person in one session, or split across roles (patient fills demographics, MA captures vitals, provider reviews)?
- Does intake need to appear as tasks in the provider's visit chart?
- Are some intake steps non-form work (e.g. lab orders, imaging) that should be modeled as orders rather than questionnaires?
- Will the same intake template be reused across visit types or programs?

Situation	Approach
Single-form intake	Skip PlanDefinition; one Questionnaire + one Bot is enough.

Situation	Approach
Multi-step / multi-author intake	`PlanDefinition` with one action per Questionnaire or ServiceRequest, applied via `$apply`; the operation creates a `CarePlan`, `RequestGroup`, and one `Task` per action.
Encounter-chart visibility	Pass `encounter` to `$apply`; each generated Task must set **both** `Task.focus` (Questionnaire/X or ServiceRequest/X – the provider UI uses this for rendering) **and** `Task.input[0].valueReference` (used to load the form). Missing either breaks the chart view.
Reusable intake template	One `PlanDefinition` with `url`, `name`, and `title` set (the provider UI's resource search uses `name`, not `title`); referenced Questionnaires and ActivityDefinitions must also have `url` so `$apply` can resolve canonicals.

Scheduling integration (appointments generated from intake answers, or intake gated on a scheduled visit) is covered in depth in the Medplum Scheduling discovery guide. From the intake side, link the resulting appointment or visit back to the intake artifacts that originated it.

3.9 Lifecycle & "Intake Complete"

Decide what "intake is done" means for your operations, how that signal is represented, and what happens to incomplete intake.

Questions:

- Is “intake complete” a single moment (form submitted) or a composite state (extraction succeeded + eligibility verified + required consents on file)?
- Which downstream workflows are gated on intake completion (scheduling, clinical, billing)?
- For incomplete intake (patient bailed mid-form, required field missing), how is that surfaced and worked?
- Are partial / draft intakes a supported state, or only completed submissions?

Situation	Approach
Simple completion	`QuestionnaireResponse.status` = `completed` is the signal; nothing further to track.
Composite completion	Single `Task` per Encounter with `businessStatus` advancing through extraction, eligibility, consent, and ready-for-clinical; or use `CarePlan.status` when intake is orchestrated via PlanDefinition (3.8).
Incomplete intake	`QuestionnaireResponse.status` = `in-progress`; surface via a Task assigned to registration so it doesn’t disappear.
Downstream gating	Scheduling, clinical chart, and billing surfaces filter on the chosen completion signal; keep the definition single-sourced.